

Architectural Frameworks for Automated Content Adaptation to Mobile Devices Based on Open-Source Technologies

Thesis submitted to the Faculty of Economics of the
European University Viadrina (Frankfurt/Oder)
in fulfillment of the requirements for the degree of
Dr. rer. pol.

Author: Bożena Jankowska

First Advisor: Prof. Dr. Eberhard Stickel
Second Advisor: Prof. Dr. Karl Kurbel

Submitted: 03.11.2006
Thesis defense: 06.09.2007

Abstract

The Web and enterprise information systems are gradually increasing their reach to a wide range of mobile devices. Although analysts hope for a breakthrough in the popularity of mobile solutions, field studies show that, except for Japan and South Korea, there is still a large gap between the technical capabilities of wireless devices/networks and the adoption of mobile services for business and private use. This paradox can be attributed to a high extent to low quality of existing mobile solutions and to their insufficient usability, represented particularly by two attributes: simplicity of use and content relevance. Additionally, network providers are afraid that mobile Internet could cannibalize their revenues from SMS and entertainment services and do not want to cooperate with service providers to improve the quality of services offered.

Wireless applications depend on device-specific features such as input/output mechanisms, screen sizes, computing resources, and support for various multimedia formats and languages. This leads to the need for multi-source authoring - the creation of separate presentations for each device type or, at least, for each class of devices. Multi-source authoring is not a cost-efficient and feasible solution, especially for mobile services consisting of numerous pages. Therefore, various single-source or device-independent techniques that try to tackle the dream of software developers for automatic content generation emerged. These techniques perform the adaptation to the characteristics of mobile devices on the client- or server-side or with the help of some intermediary, the so-called proxy. The high complexity of most adaptation frameworks, missing support for the design and development of mobile content, or the lack of non-proprietary solutions often lead to the rejection of offered adaptation methods.

This thesis describes the current state of the art in the area of adaptation techniques and proposes two new approaches for device-independent content delivery that are based on open-source technologies: the Mobile Interfaces Tag Library (MITL) framework and the Mobile Web Services Adaptation (MoWeSA) framework. The first framework is based on Mobile Interfaces Tag Library (MITL), in the second one Web Services Tag Library (WSTL) was introduced. MITL enables dynamic generation of content from existing Web pages, databases, or any other information sources. WSTL aims at the communication with Web Services. In both frameworks the development process is supported by an advanced Integrated Development Environment. Mobile services can be accessed from PCs, mobile phones or PDAs, and can be developed as browser-based or stand-alone applications. The MITL and MoWeSA frameworks currently support content delivery in the most popular formats (HTML, XHTML, WML, and Java ME) but are easily extensible to additional markup languages or standards. Tests conducted among end-user programmers confirmed that the proposed solutions are easy to learn and apply. The frameworks support the separation of tasks between page designers responsible for content presentation and programmers who modify business logic encapsulated in reusable components.

Table of contents

List of figures	IV
List of tables	VII
List of listings	IX
List of abbreviations	XI
1 Introduction	1
1.1 Problem description	1
1.2 Objectives of the work and intended contributions	5
1.3 Outline of the dissertation	6
2 Mobile infrastructure	9
2.1 Network technologies	9
2.2 Technologies for content presentation on mobile devices	17
2.2.1 Markup languages for simple user interfaces	18
2.2.2 Vocal interfaces	25
2.2.3 Mobile multimedia	29
2.2.4 Java ME technology	31
2.3 Types of mobile devices and their characteristics	39
3 Mobile content and its adoption	44
3.1 Mobile services and their application domains	44
3.2 Market for mobile devices and mobile applications	51
3.3 Adoption and acceptance of mobile Internet	55
4 Requirements and guidelines for device-independent content presentation	63
4.1 Essential functionality of adaptation and adaptation methods	63
4.2 Device Independence Principles	70
4.3 Quality model and authoring challenges for device-independent content presentation approaches	72
4.4 Design of mobile applications	81
5 Technologies for device independence	86
5.1 Methods for content adaptation	86
5.1.1 Porting the Web to mobile devices - browser-based rendering	86
5.1.2 Manual authoring and the “versionitis” problem	89
5.1.3 Scaling	94
5.1.4 Transcoding	97
5.1.5 Transforming	99
5.1.6 Comparison of methods for content adaptation	103

5.2	Categories of adaptation approaches	106
5.2.1	Intermediate adaptation	106
5.2.2	Client-side adaptation	106
5.2.3	Server-side adaptation	107
6	Related work in the area of content adaptation to mobile devices	108
6.1	Client-side adaptation approaches	108
6.1.1	CSS and related technologies	108
6.2	Intermediate adaptation approaches	112
6.2.1	Top Gun Wingman	112
6.2.2	WebAlchemist	113
6.2.3	Web Intermediaries (WBI)	114
6.2.4	Power Browser	115
6.2.5	Web Digestor and m-Links	117
6.2.6	Web clipping	119
6.2.7	WebViews	120
6.2.8	Opera Mini	121
6.3	Approaches for server-side adaptation	122
6.3.1	General technologies for server-side adaptation	122
6.3.2	XML/XSLT-based architectures	124
6.3.2.1	Underlying standards	125
6.3.2.2	Exemplary architectures based on XML/XSLT	131
6.3.3	Approaches based on User Interface Description Languages	133
6.3.3.1	Model-based User Interface Development and UIDLs	134
6.3.3.2	Related work in the area of User Interface Description Languages	135
6.3.3.3	Comparison of User Interface Description Languages	145
6.3.4	Frameworks supporting Model-View-Controller design pattern	147
6.3.4.1	Model-View-Controller design pattern	147
6.3.4.2	Exemplary MVC approaches	148
6.4	Comparison of approaches	152
7	Common elements of frameworks	156
7.1	Delivery context	156
7.2	Image converters	167
8	Mobile Interfaces Tag Library (MITL) framework	171
8.1	Design objectives	172
8.2	Evolution of architectures – from early programs to multi-tier systems	174
8.3	Framework's architecture and open-source technologies used	176
8.3.1	General architecture	176
8.3.2	JSP tag libraries	178

8.3.3	JavaServer Pages Standard Tag Library	186
8.3.4	Database access	190
8.3.5	Wrappers	193
8.3.6	Formats for Web feeds - RSS and Atom	196
8.4	MITL tags	199
8.5	Integrated Development Environment for MITL	224
8.6	Usage examples	228
8.7	Evaluation of the framework	242
9	Mobile Web Services Adaptation framework	247
9.1	From components to Service-Oriented Architectures	248
9.2	Basic technologies for Web Services	251
9.3	Mobile Web Services	260
9.4	Design objectives	265
9.5	Architecture of Mobile Web Services Adaptation framework, Web Services Tag Library	266
9.6	MoWeSA IDE	271
9.7	Usage examples	272
9.8	Evaluation of the approach	282
10	Conclusions and future work	284
10.1	Summary and key theses of the dissertation	284
10.2	Further research and development possibilities	289
	Bibliography	299
	Appendix 1: Examples of phones from various classes of mobile devices	333
	Appendix 2: ISO/IEC 9126 base measures	337
	Appendix 3: CSS producing output similar to SSR technology	340
	Appendix 4: UAProf for Nokia 7210	342
	Appendix 5: Questionnaire for the evaluation of MITL-based framework	347
	Appendix 6: Questionnaire for the evaluation of MoWeSA framework	353

List of figures

Figure 2.1: Evolution path of network generations _____	11
Figure 2.2: Evolution of markup languages _____	18
Figure 2.3: WAP architecture _____	21
Figure 2.4: Structure of WML language _____	21
Figure 2.5: Exemplary WML cards displayed on Openwave V7 Simulator _____	22
Figure 2.6: Architecture of VoiceXML applications _____	26
Figure 2.7: Environments within the Java Platform _____	32
Figure 2.8: MIDP User Interface classes _____	35
Figure 2.9: Over The Air (OTA) provisioning _____	37
Figure 2.10: Mobile keyboards _____	42
Figure 2.11: Text input possibilities _____	43
Figure 3.1: Mobile business and mobile commerce _____	46
Figure 3.2: Development of mobile services _____	47
Figure 3.3: Mobile, Internet, and fixed telephony subscriptions _____	51
Figure 3.4: Expected global mobile service revenue in 2008 _____	54
Figure 3.5: Deployment and usage of mobile technologies for mobile data access _____	55
Figure 3.6: Usage of mobile information services _____	58
Figure 3.7: Potential problems with the use of data services _____	60
Figure 3.8: Reasons for not using mobile Internet _____	61
Figure 4.1: Widow/orphan control pagination _____	66
Figure 4.2: Sectioning pagination technique _____	67
Figure 4.3: Regions pagination technique _____	67
Figure 4.4: Quality in the lifecycle _____	72
Figure 4.5: ISO 9126 quality standard _____	74
Figure 5.1: Exemplary HTML pages rendered by microbrowsers _____	89
Figure 5.2: Exemplary mobile pages _____	93
Figure 5.3: Visualization techniques in scaling _____	95
Figure 5.4: Results of MobileGoogle transcoding for Sun's Java technology page _____	98
Figure 5.5: Intermediate adaptation _____	106
Figure 5.6: Client-based adaptation _____	107
Figure 5.7: Server-based adaptation _____	107
Figure 6.1: Small-Screen Rendering for Nokia.co.uk page _____	111
Figure 6.2: WebAlchemist transcoding system _____	113
Figure 6.3: Web Intermediaries _____	115
Figure 6.4: Power Browser proxy _____	116
Figure 6.5: Power Browser navigation methods _____	116
Figure 6.6: m-Links interface on NeoPoint 1000 _____	118
Figure 6.7: m-Links architecture _____	119
Figure 6.8: Web clipping architecture _____	120

Figure 6.9: WebViews architecture	120
Figure 6.10: Sun's Java page displayed in Opera Mini browser	122
Figure 6.11: Converting XML data to different output formats	130
Figure 6.12: Request processing in Cocoon	132
Figure 6.13: Dynamic generation of XSLT style sheets	133
Figure 6.14: Meta-interface model of UIML	136
Figure 6.15: Basic structure of XIIML	138
Figure 6.16: XIIML elements	139
Figure 6.17: Cameleon reference framework (USIXML)	139
Figure 6.18: Simplified MyXML process	141
Figure 6.19: DDL syntax	142
Figure 6.20: DDL adaptation framework	143
Figure 6.21: Architecture of RIML adaptation engine	145
Figure 6.22: MVC design pattern	147
Figure 6.23: Struts overview	148
Figure 6.24: UML class diagram representing JSF components	149
Figure 6.25: Design-time application generation in MDAT	150
Figure 6.26: MORE architecture	151
Figure 7.1: Main JIMI classes	170
Figure 8.1: Mobile devices supporting WML and XHTML	173
Figure 8.2: Architecture of MITL framework	176
Figure 8.3: Request processing in MITL framework	177
Figure 8.4: Tag lifecycle	182
Figure 8.5: IterationTag lifecycle	185
Figure 8.6: BodyTag lifecycle (JSP 1.2)	185
Figure 8.7: Result of transformation with XML processing library	188
Figure 8.8: JDBC mechanisms for communication with databases	191
Figure 8.9: Processing model in Web Language	195
Figure 8.10: Request processing with MITL	200
Figure 8.11: Results produced by PaginationTag (for a MIDlet)	213
Figure 8.12: Index generated by PagerTag library	215
Figure 8.13: Table generated by MITL library (Java ME application)	217
Figure 8.14: Timetable and its transformation	223
Figure 8.15: Lengthways list and its transformation	223
Figure 8.16: Integrated Development Environment for MITL	226
Figure 8.17: Shortcuts from the taskbar of the IDE for MITL	227
Figure 8.18: Mobile Google search service in XHTML (Nokia 6600)	232
Figure 8.19: Mobile Google search service as Java ME application (Java ME simulator)	232
Figure 8.20: Mobile Google search service in HTML (Internet Explorer)	233
Figure 8.21: Results of the search with Google Tag Library	234
Figure 8.22: XHTML presented in Siemens SX1 browser	236

Figure 8.23: WML displayed in Motorola C975 browser _____	236
Figure 8.24: HTML version of weather forecasting service (in Internet Explorer) _____	237
Figure 8.25: Yahoo! Weather forecasting service on Nokia 7650 (XHTML version) _____	240
Figure 8.26: Yahoo! Weather forecasting service on Sony Ericsson T610 (WML version) _____	241
Figure 8.27: Yahoo! Weather forecasting service displayed in Internet Explorer _____	241
Figure 9.1: UML description of SOA _____	250
Figure 9.2: Web Services programming stack _____	252
Figure 9.3: Structure of SOAP and a sample SOAP response _____	255
Figure 9.4: WSDL structure _____	256
Figure 9.5: Basic UDDI data structures _____	257
Figure 9.6: Main steps in the Web Services paradigm _____	258
Figure 9.7: Communication process in an RPC-based Web Service _____	259
Figure 9.8: Communication process in a document-based Web Service _____	259
Figure 9.9: Invocation of RPC-based Web Services in Java ME _____	262
Figure 9.10: MoWeSA architecture _____	267
Figure 9.11: Façade Pattern _____	268
Figure 9.12: IDE for MoWeSA _____	271
Figure 9.13: Mobile Kayak service in WML (Sony Ericsson T610) _____	275
Figure 9.14: Mobile Kayak service in XHTML (Nokia 6600) _____	276
Figure 9.15: Enterprise Data Web Service in WML (Nokia 6610) and XHTML (Siemens SX1) _____	282
Figure 9.16: Enterprise Data Web Service as Java ME application _____	282
Figure 10.1: Web 2.0 _____	295
Figure 10.2: Traditional Web application model and Ajax Web application model _____	297

List of tables

Table 2.1: Comparison of mobile network generations _____	10
Table 2.2: Overview of WLAN standards _____	14
Table 2.3: Elements of XHTML MP _____	20
Table 2.4: Comparison of WML and XHTML MP + WCSS _____	22
Table 2.5: Overview of Java Platform Micro Edition configurations _____	33
Table 2.6: MIDlet packaging attributes _____	34
Table 2.7: MIDP UI components _____	36
Table 2.8: WAP standards (WML, XHTML) vs. Java ME _____	38
Table 2.9: Types of mobile devices _____	40
Table 3.1: Mobile enterprise applications _____	48
Table 3.2: Application trends in mobile business _____	48
Table 3.3: Opportunities and benefits of mobile applications for employees _____	49
Table 3.4: Relative monthly consumption of mobile data services as of December 2005 _____	53
Table 3.5: Percentage of companies where mobile workers have access to mobile data applications _____	54
Table 4.1: Exemplary context of use attributes _____	73
Table 4.2: Usability criteria in QUIM _____	77
Table 4.3: Relationship between factors and criteria (subfactors) in QUIM _____	78
Table 4.4: Device Independence Authoring Challenges (DIAC) _____	81
Table 4.5: Mobile Web best practices _____	85
Table 5.1: Evaluation of content adaptation methods with regard to DIPs _____	104
Table 5.2: Comparison of content adaptation methods with the help of DIACs _____	105
Table 6.1: Media types for CSS _____	109
Table 6.2: Comparison of UIDLs _____	146
Table 6.3: Evaluation of content adaptation methods with regard to DIPs _____	152
Table 6.4: Comparison of content adaptation methods with the help of DIACs _____	154
Table 6.5: Comparison of content adaptation methods with the help of DIACs _____	155
Table 7.1: Sample HTTP headers _____	158
Table 7.2: Common MIME types _____	159
Table 7.3: UAProf attributes/properties - Hardware Platform _____	162
Table 7.4: UAProf attributes/properties - Software Platform _____	162
Table 7.5: UAProf attributes/properties – Browser User Agent _____	163
Table 7.6: Web Services methods for querying and changing UPS profiles _____	165
Table 7.7: Java libraries for image manipulation _____	169
Table 8.1: Elements in tag library descriptor _____	181
Table 8.2: Tag and BodyTag interface methods _____	181
Table 8.3: Return codes for tags _____	184
Table 8.4: JSP Standard Tag Library _____	186
Table 8.5: Basic tags from core JSTL library _____	187
Table 8.6: Tags from XML processing library _____	187

Table 8.7: Basic tags from formatting library _____	190
Table 8.8: Tags for interactions with databases _____	192
Table 8.9: Wrapper toolkits _____	194
Table 8.10: Web Language modules _____	195
Table 8.11: Basic differences between Atom 1.0 and RSS 2.0 _____	197
Table 8.12: Java-based RSS/Atom libraries _____	199
Table 8.13: Design principles for mobile CSS _____	203
Table 8.14: Command type constants _____	213
Table 8.15: Tags and attributes in PagerTag library _____	214
Table 8.16: ImageItem layout constants _____	218
Table 8.17: Tags in ImageTag library _____	219
Table 8.18: Search query parameters in Google _____	228
Table 8.19: Tags from Google Tag Library _____	234
Table 8.20: Evaluation of MITL and IDE _____	244
Table 8.21: Evaluation of MITL with the help of Device Independence Authoring Challenges _____	245
Table 9.1: Comparison between RPC and document style for Web Services _____	253
Table 9.2: Existing Java XML parsers for mobile devices _____	261
Table 9.3: Java data types for JAX-RPC method arguments _____	264
Table 9.4: Selected functions from Kayak Web API _____	273
Table 9.5: Evaluation of the MoWeSA framework and its IDE _____	283
Table 10.1: SHDM artifacts _____	293

List of listings

Listing 2.1: Simple WML document	23
Listing 2.2: JSP page processing WML	23
Listing 2.3: Exemplary VoiceXML document	27
Listing 2.4: Simple SALT application	28
Listing 2.5: Exemplary MMS message	31
Listing 2.6: Exemplary MIDlet code	38
Listing 4.1: Layout specification in SMIL	65
Listing 5.1: Exemplary RDF document	101
Listing 6.1: Example of media types	110
Listing 6.2: Media queries	110
Listing 6.3: Sample XML document (author.xml)	125
Listing 6.4: DTD for author.xml document (author.dtd)	126
Listing 6.5: XML Schema for author.xml document	127
Listing 6.6: XSLT stylesheet for author.xml	129
Listing 6.7: XLink/XPointer example	130
Listing 6.8: Simple UIML document	137
Listing 6.9: WML code generated from UIML document	137
Listing 7.1: Example of Accept headers	157
Listing 7.2: Retrieving HTTP headers in Java	159
Listing 7.3: Fragment of CC/PP profile serialized to XML	160
Listing 7.4: Fragment of WURFL profile for Nokia 7110	164
Listing 7.5: Retrieving CC/PP attributes with DELI	166
Listing 8.1: RandomVal tag handler	179
Listing 8.2: TLD file for the RandomVal class	180
Listing 8.3: Web application deployment descriptor	180
Listing 8.4: Exemplary usage of core JSTL library	187
Listing 8.5: XML file (students.xml)	188
Listing 8.6: XSLT file (transform.xsl)	189
Listing 8.7: Example of a JSP page with tags from XML processing library	189
Listing 8.8: Example of DBTags usage	193
Listing 8.9: Web Language example	196
Listing 8.10: Exemplary RSS 2.0 feed for Google	197
Listing 8.11: Atom feed	198
Listing 8.12: Relevant fragments of DocTag	209
Listing 8.13: CSS file generated automatically for Nokia 7210	210
Listing 8.14: Java ME skeleton	211
Listing 8.15: Generation of Altavista index with PagerTag library	215
Listing 8.16: Use of Jakarta's ImageTag library	219
Listing 8.17: Exemplary WebL script for extracting data from Google search results	229

Listing 8.18: XML generated by the WebL script Google.webl _____	230
Listing 8.19: Mobile Google search service (google.jsp) _____	230
Listing 8.20: Mobile Google search service (rsltGoogle.jsp) _____	231
Listing 8.21: Code for Google search with MITL and GoogleTag Library _____	235
Listing 8.22: Fragments of code for the interaction with database _____	236
Listing 8.23: RSS feed for Yahoo! Weather _____	239
Listing 8.24: Extracting information from RSS feed with RSS Utilities tag library _____	240
Listing 8.25: Retrieving forecasts with XML processing library _____	240
Listing 9.1: Example of RPC-style message _____	254
Listing 9.2: Example of document-style message _____	254
Listing 9.3: XML results of hotel search _____	274
Listing 9.4: Code of the start page of mobile Kayak service (kayak.jsp) _____	274
Listing 9.5: Document displaying the results of Kayak's search query (rsltKayak.jsp) _____	275
Listing 9.6: Product class (Product.java) _____	277
Listing 9.7: Supplier class (Supplier.java) _____	278
Listing 9.8: SupplierInfo class (SupplierInfo.java) _____	278
Listing 9.9: WSDL file _____	279
Listing 9.10: SOAP request in Enterprise Data Web Service _____	280
Listing 9.11: SOAP response in Enterprise Data Web Service _____	280
Listing 9.12: Welcome page for Enterprise Data Web Service _____	281
Listing 9.13: Page with results of EDWS _____	281

List of abbreviations

1xEV-DO	One Carrier Evolved Data Optimized
1xEV-DV	One Carrier Evolved Data and Voice
1xRTT	One Carrier Radio Transmission Technology
3GPP	Third Generation Partnership Project
AAIML	Alternate Abstract Interface Markup Language
AE	Adaptation Engine
AIO	Abstract Interaction Object
AJAX	Asynchronous JavaScript and XML
AMPS	Advanced Mobile Phone System
API	Application Programming Interface
ARPU	Average Revenue per User
ASR	Automatic Speech Recognition
AST	Abstract Syntax Tree
AUML	Abstract User Markup Language
B, b	B - Byte, b - bit
BMP	Windows Bitmap
CAMELEON	Context Aware Modeling for Enabling and Leveraging Effective interaction
CC/PP	Composite Capabilities/Preference Profile
CDC	Connected Device Configuration
CDMA	Code Division Multiple Access
CEPT	Conférence européenne des administrations des postes et des télécommunications
cHTML	Compact Hypertext Markup Language
CIO	Concrete Interaction Object
CIO	Chief Information Officer
CLDC	Connected Limited Device Configuration
CML	Compressed Markup Language
CMYK	Cyan, Magenta, Yellow, Black
CONSENSUS	3G Mobile Context Sensitive Adaptability - User Friendly Mobile Work Place for Seamless Enterprise Applications
CRUD	Create, Retrieve, Update, and Delete
CSS	Cascading Style Sheets
CUI	Concrete User Interface
CUR	Windows Cursor
DBMS	Database Management System
DDL	Dialog Description Language

DECT	Digitally Enhanced Cordless Telephony
DELI	Delivery Context Library for CC/PP and UAProf
DevInf	SyncML Device Information
DIAC	Device Independence Authoring Challenge
DICOM	Digital Imaging and Communications in Medicine Format Bitmap
DIP	Device Independence Principle
DISL	Dialog and Interface Specification Language
DIWE	Device-Independent Web Engineering
DOM	Document Object Model
DPS	Digital Signal Processor
DRAM	Dynamic Random Access Memory
DTD	Document Type Definition
DTMF	Dual Tone Multi Frequency
EDGE	Enhanced Data Rates for Global Evolution
EPS	Encapsulated PostScript
ETSI	European Telecommunications Standards Institute
FITS	Flexible Image Transport System Graphic
FO	Formatting Objects
FTP	File Transfer Protocol
FUI	Final User Interface
Gb	Gigabit
GFC	Generic Connection Framework
GIF	Graphics Interchange Format
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HDML	Handheld Device Markup Language
HDTP	Handheld Device Transport Protocol
HIPERLAN	High Performance Radio Local Area Network
HIPERMAN	High Performance Radio Metropolitan Area Network
HomeRF	Home Radio Frequency
HomeRF WG	Home Radio Frequency Working Group
HSCSD	High Speed Circuit Switched Data
HSDPA	High Speed Downlink Packet Access
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
ICO	Windows Icon

IDE	Integrated Development Environment
IEC	International Engineering Consortium
IETF	Internet Engineering Task Force
IFF	Interchange File Format
IMT-2000	International Mobile Telecommunications 2000
IP	Internet Protocol
IRMA	Information Resources Management Association
IS	Interim Standard / Information System
ISM	Industrial-Specific-Medical
ISO	International Organization for Standardization
ITTP	Intelligent Terminal Transfer Protocol
ITU	International Telecommunications Union
JAD	Java Application Descriptor
JAM	Java Application Manager
JAR	Java Archive
Java EE	Java Platform, Enterprise Edition
Java ME	Java Platform, Micro Edition
Java SE	Java Platform, Standard Edition
JAXB	Java Architecture for XML Binding
JAXM	Java API for XML-based Messaging
JAXP	Java API for XML Processing
JAXR	Java API for XML Registries
JAX-RPC	Java API for XML-based RPC
JAX-WS	Java API for XML Web Services
JCP	Java Community Process
JDBC	Java Database Connectivity
JDK	Java Development Kit
JNI	Java Native Interface
JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
JSF	JavaServer Faces
JSP	JavaServer Pages
JSR	Java Specification Request
JSTL	JavaServer Pages Standard Tag Library
JVM	Java Virtual Machine
Kbps	Kilobits per second
LAN	Local Area Network
LW	Logical Window

MANNA	Map Annotations Assistant
MB	Megabyte
MDAT	Multi-Device Authoring Technology
MExE	Mobile Execution Environment
MID	Mobile Information Device
MIDlet	MIDP Application
MIDP	Mobile Information Device Profile
MIPS	Mega Instructions per Second
MIRS	Multimodal Interaction and Rendering System
MITL	Mobile Interfaces Tag Library
MMC	Multimedia Memory Cards
MMS	Multimedia Messaging Service
MONA	Mobile Multimodal Next Generation Applications
MORE	Multichannel Object REnderer
MoWeSA	Mobile Web Services Adaptation framework
MUG	Microsoft Usability Guidelines
MUID	Model-based User Interface Development
MVC	Model-View-Controller
NAICS	North American Industry Classification System
NLP	Natural language Processing
NMT	Nordic Mobile Telephone
NTSC	National Television System Committee
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open Database Connectivity
OMA	Open Mobile Alliance
OPML	Outline Processor Markup Language
OSD	Open Source Definition
OWL	Web Ontology Language
PAL	Phase Alternating Line
PBM	UNIX Portable Bitmap Graphic
PCD	Photo-CD Image
PCS	Personal Communication Services
PDA	Personal Digital Assistant
PDC	Personal Digital Communications
PDF	Portable Document Format
PGM	Portable Graymap Graphic
PIMA	Platform-Independent Model for Applications
PNG	Portable Network Graphics

PNM	Portable Any Map Graphic Bitmap
PPM	Portable Pixmap Graphic
PQA	Palm Query Application
PS	PostScript
PSD	Photoshop Format
PSTN	Public Switched Telephone Network
PU	Presentation Unit
QoS	Quality of Service
QUIM	Quality in Use Integrated Measurement
QVGA	One Quarter Video Graphics Array
RAM	Random Access Memory
RAS	Sun Raster Graphic
RCP	Rich Client Platform
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RFC	Request for Comments
RGB	Red, Green, Blue
RI	Reference Implementation
RIML	Renderer Independent Markup Language
RMS	Record Management System
ROM	Read-Only Memory
RPC	Remote Procedure Calling
RSS	Really Simple Syndication/Rich Site Summary/RDF Site Summary
SAAJ	SOAP with Attachments API for Java
SADiC	Semantic API for the Delivery Context
SALT	Speech Application Language Tags
SD	Secure Digital
SDAZ	Speed-Dependent Automatic Zooming
SDK	Software Development Kit
SDRAM	Synchronous DRAM
SGML	Standard Generalized Markup Language
SHDM	Semantic Hypermedia Design Method
SIM	Subscriber Identity Module
SMB	Smart Bookmark
SMIL	Synchronized Multimedia Integration Language
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol

SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SoC	Separation of Concerns
SQL	Structured Query Language
SSR	Small-Screen Rendering
STU	Semantic Textual Unit
SVG	Scalable Vector Graphics
SWAP	Shared Wireless Access Protocol
SWT	Standard Widget Toolkit
TACC	Transformation, Aggregation, Caching, Customization
TACS	Total Access Communications System
TCO	Total Cost of Ownership
TDMA	Time Division Multiple Access
TF/IDF	Term Frequency-Inverse Document Frequency
TFT	Thin Film Transistor
TGA	TrueVision Targa Graphic
TIFF	Tagged Image File Format
TRaX	Transformation API for XML
TTM	Time to Market
TTML	Tagged Text Markup Language
TTS	Text-To-Speech Synthesis
UAProf	User Agent Profile
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
UI	User Interface
UIDL	User Interface Description Language
UIML	User Interface Markup Language
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
UNSPSC	Universal Standard Products and Services Code System
UPnP	Universal Plug and Play
UPS	Universal Profiling Schema
URI	Unified Resource Identifier
URL	Unified Resource Locator
USIM	UMTS Subscriber Identity Module
USIXML	USer Interface eXtensible Markup Language
VAQUITA	reVerse engineering of Applications by Questions, Information and Transformation Alternatives

VGA	Video Graphics Array
VLSI	Very Large-Scale Integrated (circuit)
VoiceXML	Voice Extensible Markup Language
W3C	World Wide Web Consortium
WAE	Wireless Application Environment
WALL	Wireless Abstraction Library
WAP	Wireless Application Protocol
WBI	Web Intermediaries
WBMP	Wireless BitMaP
WCA	Web Clipping Application
WCDMA	Wideband Code Division Multiple Access
WCSS	Wireless Cascading Style Sheets
WDP	Wireless Datagram Protocol
WEST	Web Browser for Small Terminals
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WPAN	Wireless Personal Area Network
WSDL	Web Services Description Language
WSP	Wireless Session Protocol
WSTL	Web Services Tag Library
WTA	Wireless Telephony Application
WTLSP	Wireless Transport Layer Security Protocol
WTP	Wireless Transaction Protocol
WURFL	Wireless Universal Resource File
WUSA	Wireless Usability Software Agent
WWW	World Wide Web
XBM	X Bitmap Graphic
XGA	eXtended Graphics Array
XHTML	eXtensible HyperText Markup Language
XIML	eXtensible Interface Markup Language
XML	eXtensible Markup Language
XPM	XPixmap
XSLT	eXtensible Stylesheet Language Transformation
XUL	eXtensible User Interface Language

1

Introduction

1.1 Problem description

The World Wide Web, introduced by Tim Berners-Lee in 1989, has considerably lowered the threshold for information access for private and business users¹ and encouraged them to share data. It has also influenced the way of exchanging data between industry partners and employees within an organization. At the beginning of the Internet era, content could be viewed primarily on one type of device - desktop computers equipped with a browser supporting HyperText Markup Language [W3C99a]. Although the Web was envisioned as a place for free data sharing, the creators of browsers found ways to use it to their advantage and to increase their market shares. They introduced novel and unique features to their products and exercised control over the presentation of information by rendering it in a slightly different manner. These so-called “browser wars” resulted in compatibility problems. They forced Web developers to test their applications using various browsers to avoid potential troubles with unsatisfied users [Wind04]. The problem of adapting Web content to the characteristics of browsers was never seriously tackled by researchers. It was not considered as an issue that could influence the acceptance of the Internet, although it surely contributed to noteworthy shifts in the popularity of browsers. The differences in capabilities of computers like varying screen sizes, diverse resolutions, or incongruent graphical features were of marginal importance. Web pages could be adjusted to them with the help of appropriate client-side scripts.

The need for content adaptation to the features of end-devices or, more generally, for the development of context-aware applications was a consequence of technological advances that enabled access to information from sundry mobile devices. With the advent of the mobile computing era, it was no longer possible to deliver the same or slightly modified content to all devices. The amount, presentation, and types of data displayed on different handsets should differ depending on the device category and additional factors, usually summarized under the notion of context. Context is defined as “any information that can be used to characterize the situation of an entity, where an entity can be a person, place, physical or computational object” [Dey01]. Contextual information refers not only to device characteristics but also to all circumstances in which a computing task takes place. Context-aware systems, therefore, should respond to context changes and provide users with context-sensitive information, depending on the users’ tasks, location, available resources, etc.

¹ The popularity of the Internet is reflected by the statistics of its usage: in November 2005, 15 percent of the worldwide population (973 million users) was “online” and the penetration rates in North America, Australia and in some European countries reached up to 69% [IWS05]. 11.5 billion Web pages are currently available and this number is continuously rising [GuSi05].

The greatest challenge for content adaptation approaches are device features. The main problems faced in the development of user interfaces for wireless devices are [Mall03; Tara02a]:

- different input and output mechanisms
- limited resources of devices and unstable network connections
- different layout facilities
- varying screen sizes
- plethora of supported languages and formats.

Early research on adaptation approaches anticipated the expected but unrealized evolution of the wireless sector. The situation started to change with the introduction of networks with better transmission rates and new customer-friendly billing models. After their initial failure, mobile applications slowly regained their popularity and the importance of adaptation approaches increased. In 2005, mobile subscriptions approached 2.2 billion worldwide and, with an average growth rate of 9 percent per year, 3 billion are expected to be achieved by 2009 [Drak+05]. Voice and SMS currently remain the main sources of revenues [Delo05; Litt04], but the telecommunication industry is slowly switching its focus to mobile Internet.

With growing capabilities and falling prices for various types of mobile devices and services, the needs and expectations of consumers regarding availability and consumption of information evolve gradually. Hopes that mobile business applications will deliver future profit seem to be justified: many companies started to adopt mobile solutions faster than expected because of increasing competitive pressures. Business usage mainly focuses on applications that can increase the agility of enterprises. Mobile line-of-business (LOB) applications such as logistics, field service, and sales force programs are enjoying growing recognition [BenM03; DaZeGr05]. Further application areas are messaging, e-mail, access to an ERP system's functionality, viewing Intranet content, checking typical Web information such as news or stock prices, and communicating with different databases [ATKe+05; Lehn02; Lehn03; Noki03]. Such services are typically delivered to handsets via the Internet or as stand-alone programs based on Java ME.

The most compelling mobile content for private users revolves around phone personalization², but mobile e-mail, messaging, and mobile Internet are also being slowly adopted. Popular mobile services include weather information, maps, sports, and national news as well as movie and entertainment listings [Metr05]. It is furthermore expected that professional users, who became familiar with wireless applications at work, will switch to mobile services for private use. The tendency to transfer wired solutions to mobile devices can already be observed, for example in the form of moblogs. Moblogs are mobile counterparts of weblogs - Web-based publications of personal thoughts and commentaries appearing periodically. The content of a moblog is posted from a mobile device. It can consist of pictures taken by a built-in mobile camera and some text entered into a form, displayed in a mobile browser or sent via e-mail/SMS [Covi05; Knud03].

² Ring tones, wallpapers, and games are willingly downloaded to mobile devices by people from different age groups [Drak+05, p. 6, Metr05].

The trend to deliver Internet content to mobile devices and to provide wireless access to enterprise information systems is supported by recent developments in the area of software architectures. With the introduction of Service-Oriented Architectures (SOAs) based on the Web Services technology, enterprises can easily extend their applications with mobile access to their own sources of information as well as to other parties' systems or databases.

Despite alluring prospects for mobile services, there is still a large gap between the technical possibilities of wireless devices or networks and the adoption of mobile services for business and private use. This paradox can be partially attributed to high transmission costs³, long download times, security questions, and inappropriate screen and keyboard sizes [cf. ATKe+04; ATKe+05; BICoDa05; Pede05]. Many users are unable to complete the transaction process because they are unfamiliar with the application or with mobile technology in general, find the navigation difficult, or do not want to pay for using inconvenient user interfaces [cf. Hors05; Pede05]. The average time for the settlement of a booking or purchasing transaction currently amounts to about 8 minutes [Hors05]. Services that can be used from desktop computers with Internet access are regarded as convenient and much cheaper substitutes for mobile services [BICoDa05]. High investment costs, inability to show customer benefits, missing standards, incompatibilities between devices, and poor quality of services are mentioned as further obstacles for using or developing mobile applications [StTe05].

While the prices for data transfers over wireless networks are decreasing and connection speeds are constantly being improved, the lack of well-designed, high-quality mobile applications that would satisfy users' expectations is one of the most important barriers in the adoption process [cf. ATKe+04; Hueb+03; Pede05]. This problem was not only noticed by researchers, but also by the organization which is responsible for the development of the Internet - the World Wide Web Consortium (W3C). With the increasing popularity of mobile services and emergence of new architectures, W3C shifted focus from markup languages and Internet protocols to Web Services, Semantic Web, and ubiquitous content access [WiMa04]. The consortium itself does not develop or suggest any specific adaptation approaches. It publishes guidelines, recommendations, and techniques that facilitate device-independent access to the content. Various initiatives like Mobile Web Best Practice Working Group [W3C05a] or Device Independence Working Group [W3C02c] were founded by the W3C. In 2005, W3C started to work on the establishment of a "mobileOK" trustmark for Web pages [W3C05]. This trustmark should indicate adherence to a set of best practice criteria for creating content displayable on mobile devices. It is of technical nature and is intended to be used by developers.

As more and more enterprises decide to reach customers or employees equipped with mobile phones, programmers have to create content targeted for diverse devices. The effort and expenses involved in the so-called multiple authoring of content are unreasonably high. Developers have to author, maintain, and revise pages intended for delivery to the increasing number of end-user devices. This leads to the commonly known "M times N problem" – an application consisting of M pages targeted at N devices

³ Current prices for mobile Internet connections vary between 0.10€ and 0.50€ per 10 kB of data transferred, depending on the provider and contract conditions.

requires M*N pages that should be created and updated. Additionally, as new devices are introduced to the market, the content has to be adapted to their features. Multiple authoring is further complicated by the fact that many applications are not created from scratch but are based on existing Web pages. The ideal solution would therefore be single authoring - creating only one version of the content for all devices.

This type of authoring is strictly related to the idea of device independence. In device-independent approaches, a single version of content is automatically adapted for a better user experience when delivered via many device types. A long-cherished dream of application developers is to create tailored mobile content automatically to achieve cost and time efficiencies. Although this vision may be very tempting, there are some essential questions that have to be addressed by approaches for automatic generation of content delivered to multiple devices. They encompass the following issues:

- a) How to express an application independently of the targeted devices?
- b) How to adapt device-independent applications to suit device-specific characteristics?
- c) How to take into account the specifics of the mobile environment, in particular the cognitive capabilities of users?
- d) How to give authors the possibility to retain some control over the final presentation of their content?
- e) How to arrange the existing Internet content to meet the capabilities of wireless devices?
- f) How to make the approaches reusable and easy to use?

Research on approaches for automatic device-independent content delivery started in the nineties and is already relatively advanced. Existing adaptation approaches try to provide at least partial answers to the mentioned questions. Ironically, the nature of existing methods represents in many cases an important barrier to the adoption process. Some of them are proprietary and not available for public use. Others are characterized by such level of complexity that content authors are discouraged from using them even before they start.

The proposed methods also do not satisfy the expectations of content developers. They usually do not allow them to exercise any control over the presentation of content. The majority of approaches cannot be applied to the existing content. The most important issue, however, concerns the usability and efficiency of the methods for content adaptation. These aspects are particularly important because building user interfaces is a time-consuming and costly process. It was estimated that software developers spend up to 50 percent of a whole project time-resources in the design, implementation, and testing of friendly and intuitive user interfaces for systems with traditional GUIs [cf. MyHuPa00]. The development of user interfaces for mobile devices is evidently even more time-consuming. Additionally, software development is currently performed to a high extent by the so-called end-user programmers. They do not have sufficient programming experience and develop applications occasionally, in an ad-hoc manner. The number of end-user programmers in the United States is expected to grow to 55 million in 2005, while the number of professional programmers will amount to 2.75 million [MyBu04].

1.2 Objectives of the work and intended contributions

This thesis has the following four general objectives resulting from the shortcomings of current solutions and missing comprehensive research in the area of content adaptation:

- to present the current state of the art in the area of mobile content and to search for reasons impeding the adoption of mobile services
- to introduce economic reasons for the development of device-independent approaches and to provide qualitative measures for the evaluation of such solutions
- to thoroughly describe the state of the art in the research on device-independent content delivery to different devices and to evaluate the offered adaptation technologies and methods, and finally
- to propose new adaptation approaches that would gain broad acceptance of authors and users due to their distinctive characteristics and could further be improved by a community of programmers interested in open-source software.

The popularity and acceptance of solutions for device-independent content delivery have increased with the introduction of more powerful devices, better transmission rates, and lower prices. Different groups are currently working on various aspects of content adaptation. Among them are non-profit organizations such as W3C [W3C02a], industry-specific consortia like Open Mobile Alliance (OMA) [OMA04], researchers at universities and experts from multiple industries, for example, from telecommunication or information systems technology. This thesis aims at providing some evidence and explanations for the augmented need and interest in device-independent content delivery. It should also ascertain further perspectives and application possibilities for adaptation approaches.

Notwithstanding high proliferation of wireless handsets, mobile applications, in particular mobile Web browsing, cannot boast of as many advocates as might be expected. The goal of this thesis is, therefore, to explore the reasons for missing adoption of mobile solutions and to examine the quality and usability of the available mobile services.

Despite significant efforts to develop adaptation methods, requirements and criteria necessary for the design and evaluation of such methods have generally been overlooked. Therefore, one of the essential goals of this research is to identify important factors that may influence the quality of adaptation techniques as perceived by developers and end-users. A general evaluation framework for adaptation approaches should be specified and subsequently applied to the assessment of proposed methods.

Device-independent content can be generated using different methods and by means of additional entities that are in charge of the adaptation process. The number of adaptation approaches is exponentially growing from year to year. Many research projects are based on previous experiments or unconsciously repeat already available techniques. A comprehensive categorization and comparison of existing methods is, however, still missing. Most works concentrate on selected groups of approaches such as methods built around the concept of User Interface Description Languages or proxy-based solutions. This thesis intends to classify offered adaptation approaches and to describe some exemplary research in the distinguished categories. Furthermore, an evaluation of selected solutions

will be given with the goal to provide support and assistance for content authors and users in the decision process.

To outperform existing techniques, adaptation approaches developed in this research should meet the following general criteria:

- easy to learn and to apply for professionals and non-experienced authors
- equipped with a visual Integrated Development Environment (IDE) for quick and intuitive development of device-independent applications⁴
- reusable and based on non-proprietary software to guarantee widespread adoption
- promoting division of tasks and efficient use of skills and knowledge
- extensible to new formats and languages
- based on a well-known technology
- taking into account the newest trends in the area of software architectures.

In order to satisfy these requirements, the developed approaches are designed as open-source frameworks. A formal definition of a framework states that it is a reusable, semi-complete structure that can be specialized to produce custom applications [JoFo88]. It consists of a set of components that proved to work well in other projects and can be applied by teams in different organizations [Hust+03]. Frameworks are the classic build-versus-buy proposition. If properly designed, they help to reduce the investment and development costs. According to the Linus's Law of Software Engineering, the use of open-source technology facilitates the process of further development and improvement of frameworks. This law states that "Given enough eyeballs, all bugs are shallow" suggesting that large numbers of developers are more likely to find and fix a bug before it becomes a serious problem. Additionally, source code for open-source software can be read, modified and acquired free of charge.

The final goal of this work is to identify potential improvements to proposed approaches and to indicate issues for further research in the area of device-independent content delivery to different devices.

1.3 Outline of the dissertation

This dissertation consists of ten chapters and is structured as follows: After the introduction and presentation of some important objectives of this work in chapter one, the second chapter focuses on enabling technologies for mobile computing with the goal of describing and evaluating the state of the art in this domain. This includes a short overview of four generations of mobile networks and a presentation of standards that are regarded as being complementary to wireless communication systems. These standards have potential for wide public use but the appropriate infrastructure is still not set in place for all feasible points of access. Additionally, the exploitation costs are currently impractically high for individual users, especially when compared with wired solutions⁵. Chapter two

⁴ An Integrated Development Environment can significantly increase the acceptance of a solution [cf. Desm+04].

⁵ The costs may differ depending on the provider. In Germany they currently vary between 2€ and 10€ per hour, cf. <http://www.hotspot-locations.de>. Public wireless infrastructure (hotspots) is available for free but the number of access points is still very limited.

also describes various technologies that can be applied to present content on wireless handsets. It focuses, however, only on non-proprietary standards and divides them into four subcategories: markup languages, standards for vocal and multimodal interfaces, multimedia formats and languages for building advanced user interfaces. In the last section of chapter two, a classification of different types of mobile devices according to their features is provided.

Mobile solutions were expected to be the “killer applications” but they failed first and foremost in Europe and are still not widely applied there. Chapter three focuses on economic aspects of mobile computing that are essential for this research work. They provide a justification for the development of new approaches for device-independent content adaptation. This chapter explains the basic terms associated with mobile content - which are often confused with each other - such as mobile Internet, mobile business, or mobile commerce. It presents possible application domains for mobile solutions and portrays the situation on the global market for mobile devices and content. Despite high mobile phone penetration rates, the telecommunication industry has not witnessed an uptake of advanced mobile services yet. Researchers noticed that in this case technological developments have not led to widespread adoption of available services. The most important reasons for missing acceptance of mobile solutions and possible means of improvement are described in the last part of chapter three.

Chapter four starts the second part of this research work with focus on the presentation and evaluation of current adaptation methods. It lists the requirements for device-independent content presentation approaches. Since no comprehensive framework for the quality measurement of adaptation approaches exists, diverse recommendations and good practices for the design of device-independent applications are put together to create a set of evaluation criteria. In the first section of chapter four, the notion of adaptation and its various attributes are explained. The next section presents the work of the W3C consortium on device independence principles for applications and its set of authoring challenges. Additionally, the universal ISO 9126 quality standard is introduced. The developed frameworks as well as the existing approaches are then evaluated using a selected set of attributes from these models. Last section of chapter 4 focuses on general guidelines and best practices for the design of mobile applications.

Chapter five provides a theoretical framework for the categorization of adaptation approaches. Two possible classifications are distinguished and described: adaptation with regard to methods for displaying the content and with regard to the controller of the adaptation process.

In chapter six, related work and the state of the art in the area of device-independent content adaptation are comprehensively described. The presented approaches are divided into three broad categories depending on the place of content adaptation: client-side, intermediate (proxy-based), and server-side adaptation. The last group of techniques is particularly popular and is further split into solutions relying on a combination of eXtensible Markup Language (XML) and eXtensible Stylesheet Language for Transformation (XSLT), frameworks supporting Model-View-Controller design pattern and approaches based on the concept of User Interface Description Language. Each class of solutions is illustrated through exemplary work. Subsequently, chosen representative approaches are compared with regard to device independence principles, authoring challenges, and quality criteria.

The next chapter opens the last part of the research, with innovative adaptation approaches being its main focal point. It describes two common elements of the developed frameworks - delivery context and image converters. Diverse methods and formats for retrieving the capabilities of devices, as well as their advantages and drawbacks are presented. In the following section of this chapter, some open-source image libraries are outlined and compared.

Chapter eight focuses on the first proposed adaptation framework based on Mobile Interfaces Tag Library (MITL). MITL is able to transform content to different markup languages such as Wireless Markup Language (WML), eXtensible HyperText Markup Language (XHTML) or HyperText Markup Language (HTML) and can also automatically produce Java ME applications tailored to the features of wireless devices. After a short description of the evolution in the domain of architectures, the design objectives and the architecture of the framework are introduced. Afterwards, the idea of wrappers is explained. Compaq's Web Language (WebL) that serves as a language for programming wrappers is presented. This language is then used in the framework for tailoring existing Web content to mobile devices. This chapter also describes all tags contained in the Mobile Interfaces Tag Library and some additional JSP tags used for the communication with databases, XML processing and conditional logic that are provided as part of JSP Standard Tag Library (JSTL). In the next sections, the Integrated Development Environment for MITL as well as some exemplary case studies on content adaptation, performed with the help of this approach, are discussed. Last but not least, an evaluation of the proposed solution based on surveys conducted among students and experienced programmers, is provided.

Chapter nine introduces the second approach for device-independent content delivery - the Mobile Web Services Adaptation (MoWeSA) framework based on Mobile Interfaces Tag Library and Web Services Tag Library (WSTL). This solution allows the delivery of information from different sources (including Web Services) to fat and thin mobile clients. Chapter nine explains the notion of Web Services and presents the basic technologies for them. It also outlines the existing possibilities for displaying Web Services on mobile devices and some preliminary work in this area. Similarly to chapter eight, design objectives, a detailed description of the developed solution, the advanced Integrated Development Environment for the approach, some usage examples, and an evaluation of the framework are provided.

In the last chapter of this thesis, some conclusions are drawn and issues for further research and development are discussed.

2

Mobile infrastructure

Mobile infrastructure has noticeably evolved since the early 1980s, when cellular networks were first launched. As a response to increasing capabilities of networks, manufacturers were bringing more and more powerful wireless devices to the market. The growing usage of mobile devices for voice transmission and progress in very large-scale integrated (VLSI) circuit technology forced the prices of mobile devices and mobile communication to fall considerably over a short period of time. Due to network improvements and growing capabilities of handsets, voice calls as the main application domain were gradually augmented with non-voice data, messaging, and multimedia applications. To satisfy customers' expectations and to exploit existing hardware possibilities, mobile technologies evolved from simple markup languages such as HTML or WML to more powerful and resource-intensive technologies such as Java ME or SMIL.

This chapter outlines the main components of the mobile infrastructure. Section 2.1 gives an overview of the evolution of network technologies – from the first generation of wireless networks to recent developments in this area. The subsequent section introduces the technologies for presenting content on mobile handsets. Section 2.3 classifies mobile devices according to their features.

2.1 Network technologies

The explosive growth of mobile communication and mobile services resulted mainly from growing possibilities of new generations of wireless networks. Five generations, starting with the analog cellular to the so-called fourth generation embracing different wireless access mechanisms, can be distinguished.

Figure 2.1 provides an overview of these generations and their underlying network technologies. The evolution from 1G to 4G includes technical aspects, new network architectures, and improved services. Table 2.1 gives a brief overview of existing technologies for each generation.

1G networks

First generation networks (1G) were established in the mid 1980s using the analog cellular technology. Commonly known standards for 1G are Advanced Mobile Phone System (AMPS), Nordic Mobile Telephone (NMT), and Total Access Communications System (TACS) [cf. Lehn03, pp. 28-30]. 1G systems are still in use in some countries (e.g. in the United States and Canada), but most of them have already been closed to save frequency. 1G networks were country-specific and were not able to handle growing capacity needs in a cost-efficient manner [CDMA04].

	TECHNOLOGY		FEATURES
1G	AMPS NAMPS NMT TACS	Advanced Mobile Phone Service Narrowband AMPS Nordic Mobile Telephone Total Access Communications System	- Analog voice service - No data service - Separate specifications in different countries
2G	CDMA IS-95 TDMA IS-136 GSM PDC CDPD	Code Division Multiple Access IS-95 Time Division Multiple Access IS-136 Global System for Mobile Communications Personal Digital Cellular Cellular Digital Packet Data	- Digital voice service - Speed: 9.6 Kbps to 19.2 Kbps - CDMA, TDMA, and PDC only offer one-way data transmissions - Enhanced calling features like caller ID - No always-on data connection - SIM introduced - SMS and roaming as the main services
2.5G	HSCSD GPRS EDGE CDMA2000 IS-95B and 1xRTT	High Speed Circuit Switched Data General Packet Radio Service Enhanced Data Rates for Global Evolution Code Division Multiple Access IS-95B and One Carrier Radio Transmission Technology	- Packet data communication and charging based on the volume of data - Always-on data connection - Speed up to 384 Kbps - Internet-based content accessible
3G	W-CDMA CDMA2000 family of standards (1xEV-DO, 1xEV-DV, 3x) TD-SCDMA HSDPA	Wideband Code Division Multiple Access Code Division Multiple Access 2000: One Carrier, Three Carrier Evolved Data Optimized, Evolved Data and Voice Time-Division Synchronous Code-Division Multiple-Access High Speed Downlink Packet Access	- Superior voice quality - Speed up to 2 Mbps, always-on data connection - Broadband data services like video and multimedia - Enhanced roaming - Channel-switching and packet-switching transfer - USIM introduced
4G	Standards not specified so far, they are expected in 2007		- Wireless ad hoc peer-to-peer networking - High data-rate transmission (up to 1 Gbps) - Wide coverage area, seamless roaming among different systems (ubiquitous communication)

Table 2.1: Comparison of mobile network generations [own research]

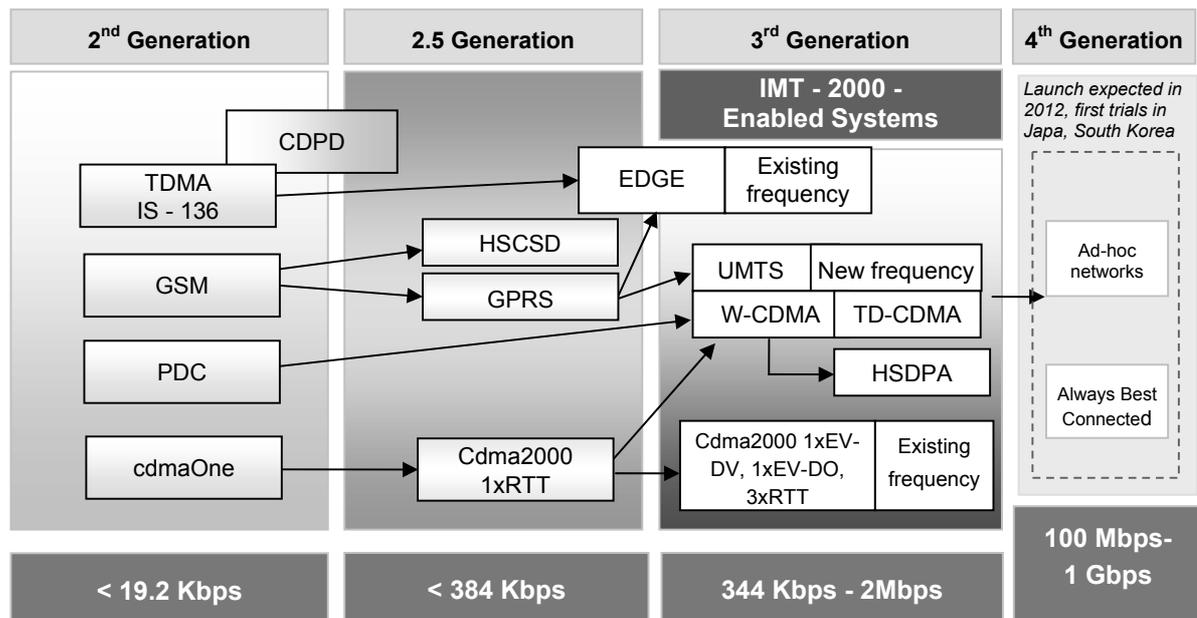


Figure 2.1: Evolution path of network generations [based on Eric02]

2G networks

Due to the limitations of 1G networks, the association of telephone and telegraph operators in Europe (the so-called Conférence européenne des administrations des postes et des télécommunications - CEPT) established a special working group called Groupe Spéciale Mobile (GSM). The goal of this group was to develop an advanced wireless system with features such as better voice quality, higher capacity, global roaming possibilities, and lower power consumption. Although a set of 2G technologies was developed, the most popular 2G standard worldwide is Global System for Mobile Communications (GSM) [cf. UMTS03]. The used standards vary across continents. In the United States, Code Division Multiple Access/Interim Standard 95 (CDMA/IS-95), called cdmaOne, and Time Division Multiple Access/Interim Standard 136 (TDMA/IS-136) are applied. Personal Digital Communications (PDC) standard prevails in Japan [Lehn03, pp. 54-62].

2G networks can achieve speeds between 9.6 Kbps and 19.2 Kbps. Voice coding, digital modulation, forward error correction, and subscriber identity module (SIM) are regarded as their major technical achievements [Jama03; SeSaPi03]. SIM, also known as a smart card, contains a computer chip and possesses a small amount of memory. Information about a subscriber, saved on the smart card, includes a telephone number, a subscriber's identity number, and the original network, to which the user is subscribed. A mobile phone reads the information from its smart card and transmits it over the network.⁶

⁶ SIM cards are not used in the 2G handsets in Japan, where data about networks and telephone numbers are integrated directly into the handset [BeWa03].

Although neglected at the beginning, the crucial service offered by 2G networks is Short Message Services (SMS), introduced in 1994 in Finland and the UK [AhKaMe+04, p. 3]. SMS is a message consisting of a maximum of 160 alphanumeric characters. It can be sent to or from a mobile device. If a message cannot be delivered immediately, because the mobile device is turned off or the user has left the coverage area, it is stored and sent to the subscriber when he/she is accessible. SMS, initially designated for personal peer-to-peer messaging, found various application domains such as interactive television (pools, game shows), dating, chatting, and push services (weather and financial news). In a study on mobile operators, Arthur D. Little forecasted that SMS will remain the main source of operators' data revenue until 2006 [Litt04, p. 39].

2G technologies have not completely eliminated the disadvantages of 1G networks. They are still not fully compatible (e.g. GSM and CDMA/IS-95 do not interoperate), and roaming between different countries may be difficult. Furthermore, these networks are optimized for voice communication and are not appropriate for data services. Low bit rates of 2G systems (9.6 Kbps for GSM) are not sufficient for services like mobile browsing or video conferencing [cf. CoSm01; PrRu03 for more details].

2.5G networks

The so-called 2.5 Generation is a transitive generation between 2G and 3G systems and embraces all advanced upgrades for 2G networks. Depending on classification, 2.5G systems usually include High Speed Circuit Switched Data (HSCSD), General Packet Radio Services (GPRS), Enhanced Data Rates for Global Evolution (EDGE), and CDMA2000.

High Speed Circuit Switched Data (HSCSD) enables theoretical data rates up to 57.6 Kbps, because a mobile station can use up to 4 time slots in parallel for a data connection. This is a relatively inexpensive way to upgrade transfer capabilities, but it has several disadvantages. The user has to pay for exploiting several time slots in a reserved channel, even if the connection is not needed (e.g. after downloading e-mails). The exploitation of more time slots per user in a circuit-switched mode leads to the reduction of channels available for voice users.

In General Packet Radio Service (GPRS), radio frequency (RF) resources are shared between circuit-switched data and voice users. GPRS-enabled networks provide higher capacity, permanent connection, and packet-based data services. The user pays for the volume of exchanged data and not for the connection time. With GPRS, theoretical transfer rates amount to 171.2 Kbps [Ande01, pp. 29-53; Pras99, p. 51].

Enhanced Data Rates for Global Evolution (EDGE) provides broadband-like data speed up to 384 Kbps. It can be used as a packet-switched enhancement for GPRS (the so-called Enhanced GPRS - EGPRS) or as a circuit-switched data enhancement (Enhanced Circuit-Switched Data - ECSD). EDGE is able to retransmit a packet that could not be delivered and supports point-to-multipoint communication [cf. Eric03].

CDMA2000 has evolved from cdmaOne. CDMA2000 IS-95B technology provides data transfer rates up to 115.2 Kbps. CDMA2000 1X radio interface is backward compatible with old standards, i.e. IS-

95A and B. It possesses an improved modulation and a better overall design, guaranteeing average bit rates of up to 144 Kbps and more capacity for voice and data services [Ande01, pp. 21-22].

3G networks

3G mobile networks, commonly known as Universal Mobile Telecommunication System (UMTS), are the next generation of mobile communication systems developed by the International Telecommunication Union (ITU). This organization defined basic requirements for 3G systems, referred to as the International Mobile Telecommunications 2000 (IMT-2000). The work of the ITU is currently continued by the Third Generation Partnership Project (3GPP). The 3GPP develops and maintains technical specification for UMTS [3GPP99; 3GPP00]. The Third Generation Partnership Project 2 (3GPP2) is responsible for the standardization process towards 3G for the Code Division Multiple Access 2000 standard [3GPP05a].

The requirements for a new generation of network systems defined by the ITU were [WaSeAl03, p. 29]:

- high data rates - from 144 Kb to 2 Mb (indoor)
- symmetrical and asymmetrical data transfer (for IP-services)
- channel-switching and packet-switching transfer
- high speech quality
- high spectrum efficiency
- seamless transition from 2nd generation systems
- global availability in all IMT-2000 networks
- independency of applications from the used net.

Wideband CDMA and CDMA2000 1x and 3x are the major UMTS contenders. W-CDMA uses the bandwidth of 5 MHz, CDMA2000 1x applies the carrier size of 1.25 MHz and CDMA2000 3x is a multi-carrier system using 3.75 MHz. A wider bandwidth means that a receiver can better distinguish incoming signals and provides improved performance. W-CDMA allocates various codes for diverse channels for voice and data transfers. It can adjust the code space every 10 milliseconds. The average transfer rates of W-CDMA amount to 220-320 Kbps, although the theoretical maximum is 2 Mbps. High Speed Downlink Packet Access (HSDPA) is an enhancement of the W-CDMA standard and can achieve peak rates up to 14 Mbps. These high rates are possible due to various techniques such as the usage of high-speed data channels, fast scheduling, higher-order modulation, or fast link adaptation [3GAm04, pp. 8-11; HoTo01; Rysa04a, pp. 20-26].

In UMTS phones, the UMTS Subscriber Identity Module (USIM) is responsible for managing security access, virus intrusion, customer profiles, and authentication. The USIM contains data about the identity of a subscriber, his/her operator, and service provider. The service profile stored on the card indicates the services that the customer has subscribed to.

3G CDMA2000 consists of 1xEV-DO (One Carrier Evolved Data Optimized), 1xEV-DV (One Carrier Evolved Data and Voice), and 3xEV-DO/DV (Three Carriers EV-DO/DV) versions. CDMA2000 1xEV-DO adds a new air interface and is able to support high data rate service on downlink (from a base station to a terminal). Separate 1.25 MHz carriers are used for the transmission of data. Up to 2.4

Mbps on downlink and 153 Kbps on uplink are possible with 1xEV-DO. CDMA2000 1xEV-DV promises data rates up to 3 Mbps and supports the transmission of data and voice over the same channel. The 3x extension of CDMA2000 uses 3 carriers instead of one [Rysa04a, pp. 46-48].

According to the UMTS Forum, worldwide subscriptions to 3G/WCDMA networks had reached 50 million by the end of January 2006. Together, WCDMA and CDMA2000 1x EV-DO networks have over 72 million users, with 25 million customers living in Europe. More than 100 WCDMA networks are operating commercially in 42 countries worldwide and over 250 devices from Asian, European, and North American manufacturers are supporting 3G networks⁷. The Freedom of Mobile Multimedia Access (FOMA) service, launched by NTT DoCoMo in 2000, surpassed 20 million in 2005 and 23.5 million are expected to be reached by the end of March 2006⁸.

Related standards

Although 2G and 3G networks dominate the mobile communication market, additional wireless technologies are used in wide and local area networks⁹. They have experienced an unexpected growth in the past few years. In the local area, the IEEE 802.11 standard with its sub-standard known as Wireless Fidelity (WiFi), and lately the IEEE 802.16 family of technologies are particularly popular. The so-called hotspot services can be found in many public areas such as restaurants, universities, airports, or hotels, providing broadband services in extremely dense urban areas. HomeRF, Bluetooth, and HIPERLAN are standards for wireless local area networks (WLAN)¹⁰. Worldwide Interoperability for Microwave Access (WiMAX) is a technology complementary to WiFi. It can be applied to wireless metropolitan area networks (MANs) because of its wide access range. Table 2.2 lists some features of the most important standards for the short-range and broadband wireless access.

Standard	IEEE-802.11 (WiFi)	HiperLAN	HomeRF	Bluetooth	WiMAX
Usage	WLAN	WLAN	Home area	WPAN	WMAN
Frequency	2.4 GHz	5.15-5.3 GHz	2.4 GHz	2.4 GHz	2-11 GHz
Maximal data speed	Up to 54 Mbps	Up to 54 Mbps	1.6-25 Mbps	Up to 2 Mbps	Up to 75 Mbps
Range	10-50 m	50-100 m	50 m	<10 m	Up to 50 km
Coverage	Hot spots	Hot spots	Hot spots	Device environment	Metropolitan areas broadband access

Table 2.2: Overview of WLAN standards [based on Lehn03, p. 138]

⁷ http://www.umts-forum.org/servlet/dycon/ztumts/umts/Live/en/umts/News_PR_Article090206.

⁸ http://www.umts-forum.org/servlet/dycon/ztumts/umts/Live/en/umts/News_3G_Article060106.

⁹ A local area network (LAN) is a network deployed in a small geographic area (e.g. an office complex or a building). A wide area network (WAN) is an arrangement of data transmission facilities that provides communication capabilities across a broad geographic area.

¹⁰ WLANs are data communication systems implemented as extensions (or as alternatives) to the wired LAN.

Bluetooth is a short-range wireless technology that provides limited-range voice and data transmission over the unlicensed 2.4 GHz Industrial-Specific-Medical (ISM) frequency band¹¹. The standard was intended to replace cables but it can also be used in LANs over short distances. Radio signals in Bluetooth are omni-directional, propagating equally in all directions. Maximally eight fixed and portable devices (one master and seven slaves, the so-called piconet) can communicate with each other using this standard. Additional 254 devices can wait in queue to join the piconet. Bluetooth has a mechanism for the discovery of services and possesses usage profiles. Devices can therefore automatically deliver available services. Bluetooth does not provide a native support for IP and is best suited for connecting PDAs, cell phones, and PCs [Ande01, pp. 81-104; Haar00; Mull02, p. 18-27; Rysa04].

Home Radio Frequency (HomeRF) is a global extension of the Digitally Enhanced Cordless Telephony (DECT) standard. DECT enables communication between different brands of cordless phones. HomeRF operates in the license-free 2.4GHz frequency band and can be applied in households for the communication between diverse consumer electronic appliances. A consortium of vendors, the Home Radio Frequency Working Group (HomeRF WG), has developed the Shared Wireless Access Protocol (SWAP) standard. It allows computers, peripherals, cordless telephones, and various electronic devices to interoperate. SWAP is able to work together with the Public Switched Telephone Network (PSTN) and the Internet and can carry voice and data traffic [Mull02, p. 152-158].

The IEEE 802.11 family of standards specifies an over-the-air interface between a wireless client and a base station or between two wireless clients. It consists of several specifications:

- 802.11 - for wireless LANs, provides 1 or 2 Mbps transmission rates in the 2.4 GHz band
- 802.11b (called Wireless Fidelity, WiFi) - supports up to 11 Mbps transmission rates in the 2.4 GHz band and has a range of up to hundred meters
- 802.11a (so called Wi-Fi5) - is an extension to 802.11, providing up to 54 Mbps in the 5 GHz band
- 802.11g - is a relatively new standard for wireless LANs and provides over 20 Mbps in the 2.4 GHz band.

It is expected that the number of Wireless LANs, based on the IEEE 802.11 family, will increase in the next years due to their flexibility, affordability, small deployment and support costs [Full03].

HIPERLAN (High Performance Radio Local Area Network) is a set of communication standards developed by the European Telecommunications Standards Institute and mainly used in Europe. Two types of HIPERLAN using 5 GHz band are known [Full03]:

- HiperLAN/1 with a maximal speed of 20 Mbps
- HiperLAN/2 with a maximal speed of 54 Mbps.

Worldwide Interoperability for Microwave Access (WiMAX), also called "WiFi on steroids", is based on the IEEE 802.16 standard (the current version is 802.16d-2004). WiMAX operates in the 2-11 GHz frequency band and should theoretically offer data transfer of up to 75 Mbps with a range of up to 50

¹¹ The exact frequency is 2402 Hz to 2480 Hz and the number of 1 MHz channels amounts to 79 [Ande01, p. 84].

kilometers. WiMAX is a wireless metropolitan area network technology and connects WiFi hotspots to the Internet providing a wireless extension to cable and DSL connections. High Performance Radio Metropolitan Area Network (HIPERMAN), created by the European Telecommunications Standards Institute (ETSI), is an alternative to WiMAX [WiFo05].

4G networks

Fourth generation networks, conceived by the Defense Advanced Research Projects Agency (DARPA), aim at the integration of existing standards and their further development. Although no clear definition of 4G exists, 4G is often referred to as an ultra high-speed wireless network with the capability to integrate existing network technologies. Two trends are particularly important for future generation networks - seamless roaming and hand-off, leading to the "Always Best Connected" (ABC) concept, and the establishment of ad-hoc networks. By 2010, planned connection speeds in 4G networks should achieve 100 Mbps. Connection speeds of 1 Gbps are seen as the ultimate goal. 4G mobile systems should be characterized by [cf. Bria+01; JaTe01; Knig05; SPGM05]:

- horizontal communication between different access technologies (e.g. wireless cellular networks with wired networks and other wireless networks such as satellite systems) in an efficient and cost-effective way
- common platform to complement other services
- connection through a common, flexible, seamless, IP-based core network
- advanced media access technology that connects the core network to different access technologies
- global roaming and interworking between different access technologies, inter- and intrasystem handover
- seamless service negotiation including mobility, security, and quality of services (perceived data rates, delay for message delivery, cost).

The next generation network infrastructure will consist of a set of partially overlapping, heterogeneous networks. Each network will provide different performance, coverage, and price. Many wireless technologies are supposed to co-exist in a heterogeneous, IP-based, wireless network. Such 4G environment will provide mobile end-users with a seamless Internet access and IP connectivity using the "best" available network. This concept is known as "Always Best Connected" (ABC) and is the central part of the 4G vision [cf. ODro+03; VaJa01].

The main problem of 4G is that different wireless networks are not really integrated and users have to interact with the system manually. This concerns the process of switching between networks, reconfiguration of a terminal, or manual re-adjustment of bandwidth utilization. Simplicity, intuitiveness, and ease of use for the end-users are the primary goals of 4G networks. The network should hide technological details from the user. It should be aware of available resources. It should select the best bearer or the combination of bearers for each task. This leads to the concept of ad-hoc networks that are formed "spontaneously", without any central administration or infrastructure [cf. Bria+01].

An important issue in 4G networks will be the Quality of Service (QoS) because of different bit rates in the supported ubiquitous infrastructure, changing bandwidth allocations, diverse fault-tolerance levels,

and hand-off¹² delays. Due to authentication procedures, requiring exchange of messages and multiple-database accesses, hand-off can be particularly problematic.

Alcatel, Nokia, Siemens, and Ericsson established the Wireless World Research Forum¹³ to work out future standards for wireless networks and to support academic and institutional research in this area [Lu02, pp. 271-357]. In 2007, telecommunications regulators should meet in Geneva to establish the first standards for 4G networks [NewF06]. Two groups are competing in this area: the first one is led by DoCoMo, Qualcomm, and European companies (e.g. Ericsson) and aims at the modification of existing technologies to improve transfer speed rates. Leadership in 4G networks was set by the Japanese government as a national goal and it supports the research and development in this area. NTT DoCoMo claims that it has already reached 1Gb transfer speeds in field tests in Yokosuka in 2003 [SPGM05]. It will be also the first company that will apply the developed technology in its products.

The second group is championed by Intel and focuses on the development of WiMax and 4G standards. It aims at developing semiconductors that will allow access to future generations of networks. The U.S. wireless carrier Sprint Nextel has already announced that it is going to spend \$3 billion in the next years to build a 4G network, basing on the technology from Intel, Samsung, and Motorola. The company assumes that the network will reach 100 million Americans by the end of 2008 [NewF06].

Many analysts are, however, quite skeptical about the 4G networks. They warn that the new generation may be a technical marvel, but users may not use it due to high user fees and availability of alternative access mechanisms like fiber optics or cable modems. Edward Snyder from Charter Equity Research represents the opinion of many other analysts claiming that "There's no business model here. Just a lot of marketing and hot air" [NewF06].

2.2 Technologies for content presentation on mobile devices

Open-source technologies for content presentation on wireless devices can generally be divided into the following four subsets:

- markup languages that merge content with presentation rules and are mainly used in wireless browsers
- technologies for vocal interfaces
- languages and formats enabling multimedia-based presentations
- technologies for the development of advanced Graphical User Interfaces (GUIs).

Additional technologies, not introduced in this thesis, include proprietary developments such as Binary Runtime for Wireless Environment (BREW, based on C or C++) from Qualcomm [Qual02], or Microsoft's eMbedded Visual Basic and Microsoft's eMbedded Visual C++ [Micr02].

¹² Hand-off refers to the process of switching devices or networks.

¹³ Cf. <http://www.wireless-world-research.org/>.

2.2.1 Markup languages for simple user interfaces

Markup languages for the creation of simple, browser-based user interfaces evolved from two standards: HyperText Markup Language (HTML) [W3C99a] and eXtensible Markup Language (XML) [W3C04b]. The evolution process is presented in figure 2.3. In Europe the most popular wireless markups are still WML and XHTML. Many browsers, especially those installed on Personal Digital Assistants and smartphones, also support HTML.

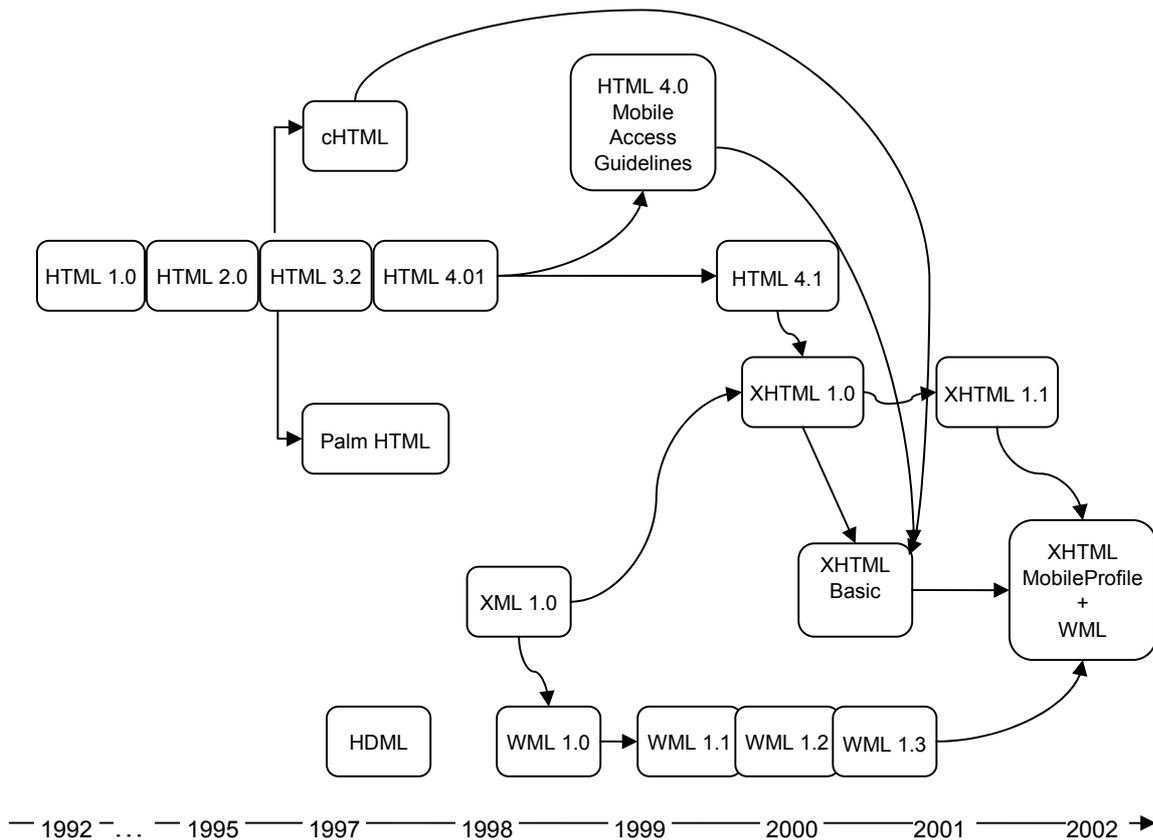


Figure 2.2: Evolution of markup languages [own research]

HyperText Markup Language (HTML)

HyperText Markup Language is called the “lingua franca” of the World Wide Web. It was originally developed by Tim Berners-Lee and popularized by the Mosaic browser. Web documents contain specific instructions (tags) that are appropriately rendered by Web browsers. HTML allows the definition of pages with different presentation elements such as paragraphs, lists, tables, or forms. The pages can be linked to other Web sites and can have embedded multimedia content. The HTML standard underwent an interesting evolution from version 1.0 to 4.1. In old versions of HTML, content, structure, and formatting instructions were placed together in a single document. The HTML specification 3.2 formalized some popular HTML extensions and acknowledged the use of tables for formatting. A straightforward consequence of this recommendation was a “tag soup”. This term refers to common authoring errors that are correctly rendered by browsers. Common mistakes include the use of presentational HTML elements to create visual effects and ignoring their semantic meaning (e.g.

the use of tables for layout), malformed or improperly nested HTML tags, unescaped character entities, etc¹⁴.

Since version 4.01, a document's structure is expected to be separated from presentation elements. Formatting instructions should be saved separately from the content in the form of Cascading Style Sheets (CSS) [W3C06]. One style sheet can be applied to diverse documents. HTML markup has to outline the structure of a document, e.g. by using the <h1> element for titles, <h2> for subtitles, etc. Furthermore, the HTML 4.01 specification recommends to resign from tables as means for layout, because non-visual media are not able to render such pages.

According to the formal HTML specification, HTML pages should use a predefined set of hierarchical, descriptive tags. This should enable the rearrangement of page structures or automatic processing of Web sites for information filtering purposes. Developers of HTML assumed that the markup will be displayable on different devices with various capabilities. This goal is, however, difficult to achieve even though the W3C provides a set of rules that should enable the deployment of HTML on mobile devices [W3C99]. Commonly known mobile HTML browsers include Pocket Internet Explorer and Opera¹⁵.

XHTML Modules

Extensible Hypertext Markup Language (XHTML) is the result of rewriting HTML 4.0 into XML. This markup language was initially designed exclusively for Internet browsers. With the introduction of a modularization concept, XHTML was split into a collection of abstract modules with specific functionalities [W3C01]. These modules can be combined with each other and eventually extended with new elements to enable content presentation on different platforms. XHTML Basic was the first XHTML subset designed for devices with limited capabilities (mobile phones, PDAs, vending machines, smart watches, etc.) [W3C00a]. It includes basic textual elements like headings, paragraphs and lists, hyperlinks, basic forms and tables, images, and meta information.

The XHTML Mobile Profile (XHTML MP) standard, developed by the Open Mobile Alliance (OMA, formerly the WAP Forum) and DoCoMo as part of the WAP 2.0 specification, combines XHTML Basic, WML, and cHTML [WAPF01]. Some elements from cHTML and WML such as `acronym`, `address`, `br`, `b`, `big`, `hr`, `i`, `small`, `dl`, `fieldset`, and `optgroup` were added to the tags from XHTML Basic. XHTML MP also supports navigation aids, onenter events, and other features from the WML model.

Table 2.3 displays the elements available in XHTML MP. In XHTML, content is separated from presentation by providing the possibility to apply Wireless Cascading Style Sheets (WCSS) – a simplified version of CSS [WAPF02a]. OMA has also specified ECMAScript-MP, a subset of JavaScript, to be used along with XHTML MP. Currently, XHTML MP together with WCSS and

¹⁴ Cf. http://en.wikipedia.org/wiki/Tag_Soup.

¹⁵ Cf. <http://www.microsoft.com/windowsmobile/pocketpc/default.msp>, <http://www.opera.com>.

ECMAScript-MP are regarded as standards for next-generation mobile browsers and are supported by most new devices.

Module	Elements
Structure	html, body, head, title
Text	abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1-h6, p, pre, q, b, i, big, samp, span, strong, var, hr, small
Hypertext	a, link
List	dl, dt, dd, ol, ul, li
Tables	caption, table, th, td, tr
Forms	form, input, label, select, option, textarea, fieldset, optgroup
Object	object, param
Images	img
Metadata	meta
Presentation	style elements and attributes

Table 2.3: Elements of XHTML MP [WAPF01a]

Wireless Markup Language (WML) and WMLScript

The Wireless Application Protocol (WAP), conceived by the WAP Forum in 1997, is an open specification that enables access to the Internet content from thin-client devices. WAP 1.x specifications include two protocols in the application layer: the Wireless Session Protocol (WSP) and the Wireless Transaction Protocol (WTP). These protocols are responsible for data exchange in wireless networks and provide functionalities similar to HTTP. The transport layer is based on the connectionless Wireless Datagram Protocol (WDP) derived from the User Datagram Protocol (UDP). Secure data transmission is guaranteed by the Wireless Transport Layer Security Protocol (WTLSP) [Fort+01, pp. 27-29].

When a mobile browser sends a request for a URI, it first goes to a mobile WAP gateway which decodes it to a plain text and forwards it to a Web server. The response of a server is sent to the WAP gateway that encodes it into the binary form of WML and sends it back to the mobile device¹⁶ [Lu02, pp. 239-260]. This process is illustrated in figure 2.3.

¹⁶ In binary encoding, tags and headers are compressed; text remains uncompressed, only all unnecessary line breaks and spaces are eliminated.

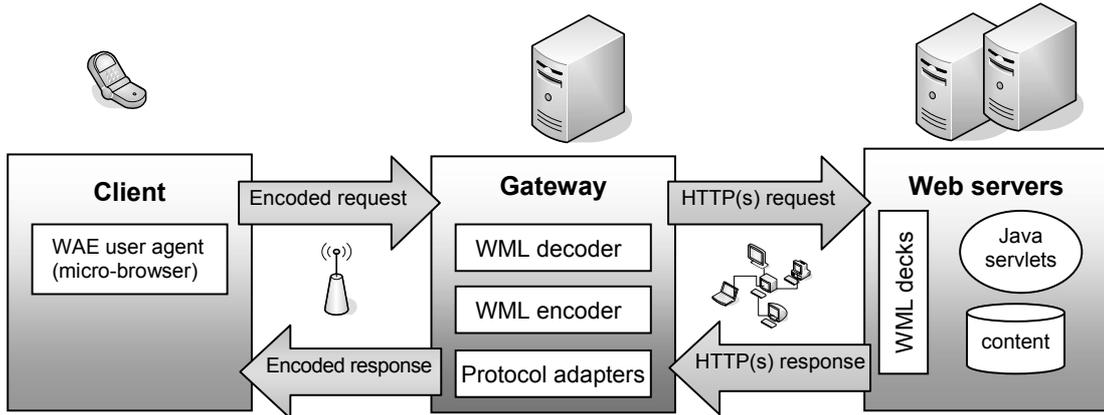


Figure 2.3: WAP architecture [Fort+01, p. 30]

In the WAP protocol stack, the upper layer is represented by the Wireless Application Environment (WAE) layer. It consists of a logical layer for user agents (browsers, phonebooks) and a layer with services and formats accessible by these user agents such as WML, WMLScript, or Wireless Telephony Application (WTA). Wireless Markup Language (WML) [WAPF02], developed for wireless devices with limited capabilities, is an application of XML. It allows displaying information, provides input options, and permits to interact with user agents (typically a browser). The basic WML document is called a deck and can contain many units of interaction and presentation, the so-called cards. WML includes a set of tags for text and image presentation, supports linking between cards and decks, and provides event handling. The elements and attributes of WML are visualized in figure 2.4. A comparison between the capabilities of WML and XHTML MP is provided in table 2.4.

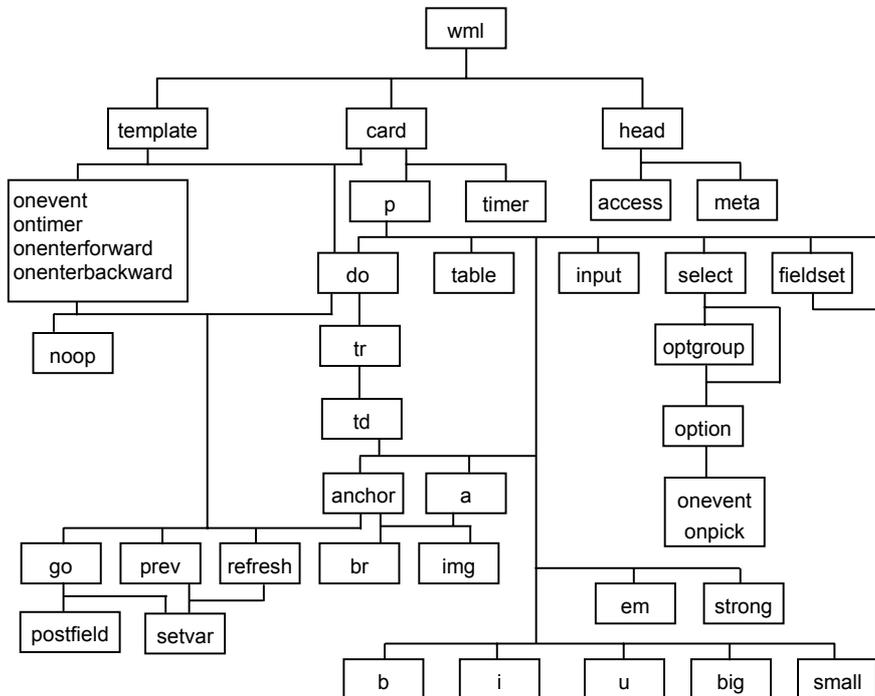


Figure 2.4: Structure of WML language [WAPF02]

Feature	WML 1.x	XHTML MP and WCSS
Standardization body	Standard developed by the WAP Forum	Standards adopted from the W3C by the OMA with a few wireless extensions
Content displaying in devices	Content and layout defined in the same document, separately tailored for each device	Content and layout defined in separate documents, the same content can have different layouts using different wireless style sheets
Content encoding	Content needs to be binary-coded	No content encoding required
Document layout control	Basic	Advanced layout control with WCSS
Color control support	Only color GIFs, but no font or background color control	Full color control support for fonts, backgrounds, borders, etc. on color devices with the help of WCSS

Table 2.4: Comparison of WML and XHTML MP + WCSS [own research]

Most WML tags are similar to these provided in HTML. Interesting new elements include `template`, `onevent`, `timer`, `do`, `setvar` and `go`, `prev`, and `postfield` tags. The `template` element defines a template with event binding. A template can be applied to all cards in a deck. The `onevent` element catches different events and specifies suitable actions. The `timer` tag generates a timer event after a given amount of time and initiates an action, such as redirecting to a new card. The `do` element gives the possibility to accomplish actions on a card (e.g. setting variables in memory, moving to another card). The `setvar` element is helpful if some variables should be stored in memory for further use. The `go` element specifies the address for passing parameters with the `GET` or `POST` method, set in the `postfield` element. The `prev` tag redirects the user to the last viewed card. Listing 2.1 shows an example of a WML document containing the mentioned elements.

The WML deck from listing 2.1 contains two cards. After a given amount of time, the user is redirected from the first card with a WBMP image to the main card (lines 8-12). On this card, he/she can enter his/her login name and password information (lines 21-22). After reloading the page, the variables are set to empty strings (lines 15-20 in listing 2.1). Subsequently, the variables are sent to the `login.jsp` page for further processing (lines 23-28). The JSP page, shown in listing 2.2, retrieves the variables from the request object and the login name is displayed on the next WML card (lines 7-9 in listing 2.2). The resulting cards are shown in figure 2.5.



Figure 2.5: Exemplary WML cards displayed on Openwave V7 Simulator

```

1. <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
2. "http://www.wapforum.org/DTD/wml_1.1.xml">
3. <wml>
4.   <template>
5.     <do type="prev" title="Last"><prev/></do>
6.   </template>
7.   <card title="Welcome" id="card1" ontimer="#main">
8.     <timer value="40"/>
9.     <p><center></center></p>
10.    <do type="accept">
11.      <go href="#main"/>
12.    </do>
13.  </card>
14.  <card title="Welcome" id="main">
15.    <onevent type="onenterbackward">
16.      <refresh>
17.        <setvar name="login" value=""/>
18.        <setvar name="password" value=""/>
19.      </refresh>
20.    </onevent>
21.    <p>Login: <input name="login"/></p>
22.    <p>Password: <input name="password" type="password"/></p>
23.    <do type="accept" label="Submit">
24.      <go href="login.jsp" method="post">
25.        <postfield name="login" value="$(login)"/>
26.        <postfield name="password" value="$(password)"/>
27.      </go>
28.    </do>
29.  </card>
30. </wml>

```

Listing 2.1: Simple WML document

```

1. <%response.setContentType("text/vnd.wap.wml");%>
2. <?xml version="1.0"?>
3. <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
4. "http://www.wapforum.org/DTD/wml_1.1.xml">
5. <wml>
6.   <card title="Result">
7.     <% String login = request.getParameter("login");
8.     String password = request.getParameter("password");
9.     out.println("<p>" + login + ", welcome on our page !</p>");%>
10.  </card>
11. </wml>

```

Listing 2.2: JSP page processing WML

For the processing of WML variables and application logic, WMLScript can be used [Fort+01, pp. 241-267]. WMLScript is able to access user agent facilities (e.g. entries in a phonebook). It can dynamically

generate dialogs and messages locally and can validate user input. WMLScript is a weakly typed language¹⁷ and contains operators, expressions, and a predefined set of functions (assembled in libraries) for String manipulation, data type conversion, creation of dialogs, URL parsing, and browser interaction. Additionally, Wireless Telephony Application (WTA) offers access to some WAP-external phone functionalities such as editing a phonebook, setting up a call, or answering it.

A set of mobile emulators and Integrated Development Environments (IDEs) is provided for WML as well as for other markup languages. These emulators enable fast prototyping of wireless applications and simulate the presentation of mobile pages on different wireless devices. Usually, one emulator is able to support various languages such as WML, XHTML, or HDML. The most commonly known emulators are delivered by Nokia, Motorola, Ericsson, and Openwave [cf. Kari03; KuDaZa02 for more details on the selection of emulators]¹⁸. Alternatively, a developer may use an online emulator. Through this thesis, emulators from different vendors are used to present mobile content.

Handheld Device Markup Language (HDML)

Handheld Device Markup Language (HDML) is a proprietary language developed by Openwave (formerly Phone.com) in 1997 [UnP99; W3C97]. It can only be viewed on wireless devices with implemented Openwave's browsers and is often considered as a predecessor of WML. HDML uses the Handheld Device Transport Protocol (HDTP) and Openwave's UP.Link gateway. A phone request is sent to the UP.Link gateway and the gateway sends it further to a Web server. The server responds with a page that is first sent to the gateway via HTTP, and the information is then sent back to the device using the HDTP protocol.

HDML applies the "deck of cards" metaphor, provides input options, and specifies actions in response to pressed keys. HDML cards can display information, prompt for input, and show lists of options. Although HDML is royalty-free and open for use, it requires support for Openwave's browsers and gateways. HDML does not allow scripting, accepts only small one-bit bitmap images (BMP) and is not based on XML.

i-mode markups

Compact HTML (cHTML) [W3C98] was developed by the Japan-based Access company for information mode (i-mode) devices. It was based on a subset of HTML extended with tags for special use on cell phones. cHTML did not support JPEG images¹⁹, image maps, tables, multiple character font types, background colors and background images, frames and style sheets. In compact HTML, all operations could be accomplished by a combination of four buttons: Cursor forward, Cursor backward, Select, and Back/Stop.

¹⁷ In weakly typed languages, data types of variables or the return types of functions do not have to be specified.

¹⁸ Nokia's emulators and software development kits (SDK) are available from <http://forum.nokia.com>, Openwave's toolkits can be downloaded from http://developer.openwave.com/dvl/tools_and_sdk/openwave_mobile_sdk.

¹⁹ GIF images are supported by cHTML.

iHTML changed its name to i-mode compatible HTML (i-HTML for simplicity)²⁰. The format was extended many times (eight versions of i-HTML exist so far) and new proprietary tags and attributes were introduced with every new specification. Additionally, NTT DoCoMo introduced i-XHTML²¹, a standard based on XHTML Mobile Profile (XHTML-MP) with some extensions. Since i-XHTML is backward compatible with old i-mode specifications, it supports most of the proprietary tags and attributes found in i-HTML. NTT DoCoMo also developed its own version of WAP CSS – i-CSS [Bov05]. I-CSS is based on a subset of CSS 2 with some extensions, but only inline styling is supported²².

Additional tags and attributes in the i-(X)HTML specification should make the interaction with wireless devices simpler. The "accesskey" attribute of the anchor tag enables faster, direct selection of anchors by using phone keys. The "tel:" attribute of the <a> tag makes it possible to open a voice connection to the telephone number provided in a link.

NTT DoCoMo also added special pictorial characters called *emoji*. Emoji are mapped to certain code points in the Unicode standard and by inserting them into a source code Web developers can easily add emoticons and icons to pages. Since there is no agreement between the Japanese carriers with regard to the mapping, different browsers display emoji as plain text or map various characters to them.

I-mode uses HTTP over TCP/IP as the protocol. Additionally, the Wireless Profiled Transmission Control Protocol (WPTCP) was developed to improve the transmission efficiency. Protocol gateways in i-mode translate WPTCP to standard TCP [Lu02, pp. 261-267; ReMa03, pp. 83-84].

2.2.2 Vocal interfaces

The explosive growth of the Internet and high penetration rates of mobile devices have forced providers of voice-based applications to offer their services through the Web. Recent advances in two technologies - Text-To-Speech Synthesis (TTS) and Automatic Speech Recognition (ASR) - enabled the development of advanced aural interfaces [JuMa00]. A computer with a Text-To-Speech Synthesis engine is able to transform any text into speech. Special tags and text serve as input to the TTS engine. It can modify the sound of the rendered voice according to the supplied tags. After the analysis of words or phrases, and resolution of possible ambiguities with the help of sentence semantics, the engine is able to synthesize text into speech. The Automatic Speech Recognition technology provides spoken input to an application by using a speech recognition engine. The engine processes spoken input and converts it into text. Three standards for vocal interfaces enjoy growing popularity: VoiceXML, SALT, and XHTML+Voice (X+V). All of them aim at supporting multimodal applications, but SALT is the most advanced standard in this domain.

²⁰ Cf. http://www.nttdocomo.co.jp/english/p_s/i/tag/imodetag.html.

²¹ Cf. http://www.nttdocomo.co.jp/english/p_s/i/xhtmll/imodetag.html.

²² Inline styling refers to the use of CSS attributes directly in tags, without a separate CSS definition.

Voice eXtensible Markup Language (VoiceXML)

Voice eXtensible Markup Language (VoiceXML), based on the XML standard, was developed by the VoiceXML Forum (a group founded by AT&T, IBM, Lucent Technologies, and Motorola) in 1999 [Abbo01; Hein03; W3C04f]. VoiceXML is designed for the development of audio dialogs that feature digitized audio, synthesized speech, recognition of spoken and Dual-Tone Multi Frequency (DTMF) key input, recording of speech, telephony and mixed-initiative conversations. VoiceXML is able to control audio output and input, presentation logic and control flow, event handling and essential telephony connections (call transfers, disconnections).

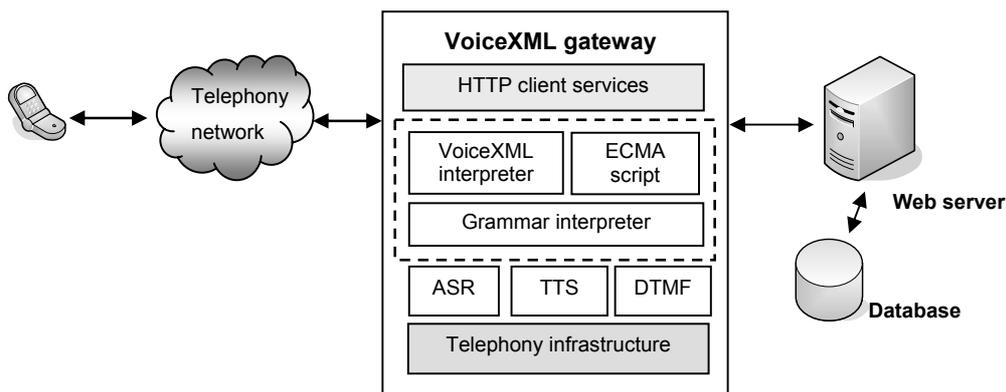


Figure 2.6: Architecture of VoiceXML applications

Figure 2.6 demonstrates the architecture of VoiceXML applications. It consists of an application server, a VoiceXML telephony server, an Internet-style network, and a telephone network. The application server is a Web server supporting the application logic. The VoiceXML telephony server interprets VoiceXML documents and is responsible for controlling speech. The Internet-style network is TCP/IP-based and connects the application server with the telephony server. The telephone network enables the connection between a mobile device and the telephony server.

VoiceXML gateway, consisting of many layers, is the most important part of the whole architecture and resides on the telephony server. The telephony services layer is responsible for telephone signaling, call transferring, and extracting relevant user information. The Automatic Speech Recognition (ASR) and DTMF layers accept user input and convert it into application-specific formats. The Text-To-Speech Synthesis (TTS) layer is in charge of producing user output. The next layer, consisting of a VoiceXML parser/processor, a grammar interpreter, and an ECMAScript interpreter, processes VoiceXML data. The HTTP client services layer fetches the information needed by applications and VoiceXML documents.

```
1. <?xml version="1.0"?>
2.   <vxml version="2.0">
3.     <menu>
4.       <prompt> Choose from <enumerate/></prompt>
5.       <choice next="sports.vxml"> sports </choice>
6.       <choice next="weather.vxml"> weather </choice>
7.       <choice next="news.vxml"> news </choice>
8.       <help>
9.         If you would like sports scores, say sports.
10.        For local weather reports, say weather, or
11.        for the latest news, say news.
12.      </help>
13.      <noinput>You must say something.</noinput>
14.      <nomatch>Please speak clearly and try again.</nomatch>
15.    </menu>
16.  </vxml>
```

Listing 2.3: Exemplary VoiceXML document

Listing 2.3 shows an exemplary VoiceXML application. The user is first confronted with a menu dialog. Menu dialogs offer lists of choices from which the user may select. Form dialogs gather inputs and display information. In the presented example, the user may choose between sport, weather, and news services. If no input is provided, an appropriate message is rendered by the TTS engine. If the ASR engine is not able to understand the selection or the user has made an incorrect choice, a message included in the `<nomatch>` tags is spoken. If the user says “help”, the message included in the `<help>` tags is provided. Answer options (in this case sports, weather, or news) may also be placed in the so-called grammar files which include possible alternatives for user input and enable a faster recognition of voice.

Speech Application Language Tags (SALT)

The Speech Application Language Tags (SALT) standard, created by the SALT Forum²³, was designed to embed voice into existing Web formats (e.g. WML, XML, HTML, SVG, or XHTML) and enables multimodal applications [SALF02]. Users interact with applications by providing information using speech, keyboard, keypad, mouse, or stylus. The output is generated as synthesized speech, audio, plain text, motion video, or graphics. Each of these modes can be applied independently or concurrently.

SALT includes a set of tags with associated attributes and Document Object Model (DOM) object properties, events, and methods. It provides DTMF and call control capabilities for browsers running voice-only applications. A typical SALT architecture consists of a Web server, a telephony server, a speech server, and a client device. The Web server generates pages containing markup language elements (e.g. from HTML or WML), SALT, and embedded scripts. These scripts control the dialog flow for voice interaction. The telephony server is responsible for connections to the telephone network

²³ The SALT Forum was founded by Cisco, Converse, Intel, Philips, Speechworks, and Microsoft (<http://www.saltforum.org>).

and incorporates a voice browser interpreting the documents. The speech server is able to recognize speech, plays audio prompts, and sends responses to the user.

Listing 2.4 shows SALT tags embedded into an HTML page. The text “This is my first multimodal application” is displayed on the page while the user is hearing the text “Hello World from SALT!”. The application only works after the installation of an appropriate plug-in for Internet Explorer that is able to recognize and to generate voice.

```
1. <html xmlns:salt="http://www.saltforum.org/2002/SALT">
2. <head>
3.   <title>SALT example</title>
4.   <object id="SpeechTags" CLASSID="clsid:33cbfc53-a7de-491a-90f3-0e782a7e347a"
5.     VIEWASTEXT></object>
6.   <?import namespace="salt" implementation="#SpeechTags"/>
7. </head>
8. <body onload="Welcome.Start()">
9.   <h3>This is my first multimodal application!</h3>
10.  <salt:prompt id="Welcome">
11.    Hello World from SALT!
12.  </salt:prompt>
13. </body>
14. </html>
```

Listing 2.4: Simple SALT application

XHTML+Voice (X+V)

XHTML+Voice (X+V) [W3C01b] adds voice interaction to traditional visual Web interfaces. XHTML and XML Events technologies are integrated with XML vocabularies developed as a part of the W3C Speech Interface Framework. X+V comprises voice modules that offer support for speech synthesis, speech dialogs, command control, and speech grammars. Event handlers that implement VoiceXML actions can be attached to XHTML elements and are able to respond to specific DOM events. CSS style sheets are in charge of the presentation, whereas XHTML reflects the structure of a document.

The modular architecture of XHTML+Voice enables the separation of an application into various components. Visual components may be programmed independently of vocal components and developed code fragments may be re-used in different applications, for example, in pure VoiceXML programs. Dividing the development between experts in particular areas gives more flexibility and guarantees higher quality of applications. Due to limited amount of content that can be displayed on small visual interfaces and increasing user demand for voice input and output, X+V is particularly well-suited for the development of wireless multimodal applications.

2.2.3 Mobile multimedia²⁴

The term multimedia refers to “the property of handling a variety of representation media in an integrated manner” [BlSt97]. Blair and Stefani [BlSt97] further differentiate between continuous media types with a temporal dimension like animations, video, voice, or audio and discrete media types like text and graphics. This categorization is also applied in this section.

Audio formats

Audio standards can generally be divided into two categories: natural and synthetic audio. In the natural audio coding, perceptual models are applied to compress natural sound. In the synthetic audio coding, sound-specific, parametric models are used to transmit sound descriptions. The descriptions are converted into sound through real-time sound synthesis [cf. Bran98].

The most popular natural audio formats for mobile devices are: Adaptive Multi-Rate (AMR), Adaptive Multi-Rate Wideband (AMR-WB), Advanced Audio Coding (AAC), Enhanced Advanced Audio Coding (EAAC+), and MPEG-1 Layer 3 (MP3) [cf. 3GPP05 for more details]. AMR, primarily used in UMTS 3G phones, allows different bit rates conveying speech, depending on the available bandwidth and signal-to-noise ratio. AMR and AMR-WB differ with regard to speech bandwidths and bit rates that are used. The AMR speech bandwidth is 3.5 kHz, whereby for AMR-WB this bandwidth amounts to 7 kHz. The bit rates for AMR are between 5 and 12 Kbps, in AMR-WB between 7 and 24 Kbps. AMR-WB+, introduced in 2004, is an extension to AMR-WB and provides higher sound quality, greater audio bandwidth, and improved bit rates (up to 48 Kbps). Advanced Audio Coding (AAC) is currently the most advanced audio coding technology used in MPEG-4. It provides high quality audio reproduction and requires approximately 50 percent less data than MP3. MPEG-1 Layer 3 (MP3) is the audio component of the MPEG-1 standard allowing for the compression of sound sequences and preserving the original level of sound quality.

The synthetic audio formats for mobile devices include different versions of Musical Instrument Digital Interface (MIDI). SP-MIDI is a specific variant of MIDI for mobile applications. It takes into consideration different devices with various numbers of musical voices.²⁵

Video and animation formats

Two organizations are currently working on the standards for video coding: the International Telecommunications Union (ITU) and the International Standardization Organization (ISO). Common Intermedia Format (CIF) and Source Input Format (SIF) are video standards that define several size formats (e.g. SQCIF 128x96 pixels and CIF 352x288). H.26x (for example, H.261, H.264) standards belong to a family of video codecs standardized by the ITU for audiovisual services targeted for video conferencing. MPEG-1, MPEG-2 and MPEG-4, MPEG-7 and MPEG-21 are international ISO standards for coding of audiovisual content developed by the Moving Pictures Expert Group (MPEG).

²⁴ Cf. also [EkHoOl01; Rasm+04].

²⁵ A good overview of audio formats is provided in [Sepp04].

The MPEG-4 (ISO/IEC 14496) standard is of particular interest to the mobile industry because it enables coding of individual audiovisual objects and possesses, in comparison to the previous formats, better compression, scalability, and error resilience. Proprietary industrial video coding standards include Windows Media Format from Microsoft, Audio Video Interleaved (AVI), and DivX from DivXNetworks.

Macromedia Flash [Macr05] is a standard applied in animations. It uses a binary format for defining shapes, frames, animations, and actions. Flash is based on vector graphics and is supported by some mobile phones and most Internet browsers.

Its main rival, Synchronized Multimedia Integration Language (SMIL), is an XML-based multimedia language [W3C05b]. In SMIL, multimedia content is divided into separate files and streams (audio, video, text, and images) that are sent to a user's device separately and then displayed together. A set of commands specifies how multimedia components should be rendered. SMIL has been adopted in multimedia players such as RealNetworks's RealOne player [Real00] and Apple's QuickTime [Appl04] and has been selected as the format for Multimedia Messaging Service (MMS) in mobile phones.

Scalable Vector Graphics (SVG) is an XML-based vector graphics format that reuses some functionality from SMIL [W3C03c]. It uses absolute positioning, interval- and event-based temporal models. Interaction is possible via links or events. For adaptation, SVG includes a switch functionality. It can be used to select and display alternative parts of a document depending on the screen size, bit rate, and selected language. Two mobile profiles of SVG 1.1 were defined using the modularization concept, also applied in XHTML. The first profile, SVG Tiny (SVGT), is suitable for cell phones. The second profile, SVG Basic (SVGB), is suitable for PDAs [W3C03b]. Several SVG players are available for PCs, PDAs, and mobile phones (e.g. TinyLine SVG Player²⁶).

Virtual Reality Modeling Language 97 (VRML) and Virtual Reality Modeling Language 2.0 (VRML2) are file formats for describing interactive 3D objects and worlds to be experienced on the World Wide Web [ISO97]. First browsers for mobile platforms with the VRML2 support are already on the market.

Messaging formats

After the unexpected success of SMS, two similar multimedia-enriched standards emerged: Enhanced Messaging Service (EMS) and its successor, Multimedia Messaging Service (MMS). An EMS-enabled mobile phone can receive and send messages that include pixel pictures and animations, sound effects, ring signals, and formatted text. Mobile Multimedia Messaging (MMS) extends EMS's capabilities and enables sending of multimedia messages from MMS-enabled devices to other mobile users and to e-mail users.

A multimedia message can include a photo or a picture postcard annotated with text and/or an audio clip, a synchronized playback of audio, a video emulating free-running presentation, or a video clip. It

²⁶ Cf. <http://www.tinyline.com/svggt/download>.

can also be a drawing combined with some text²⁷. Synchronization Multimedia Integration Language (SMIL) is used in MMS as presentation language. The introduction of MMS requires a new mobile network infrastructure (such as additional MMS gateways) that can handle multimedia content [cf. NoSv01]. A typical MMS message written in SMIL has two regions for displaying text and images, at most. The layout, synchronization and timing of MMS parts can be specified. Listing 2.5 shows an exemplary MMS message containing an image, a text, and a sound file. All elements are presented in parallel for five second.

```
1. <smil>
2. <head>
3.   <layout>
4.     <root-layout width="160" height="140"/>
5.       <region id="Image" width="160" height="120" left="0" top="0"/>
6.       <region id="Text" width="160" height="20" left="0" top="120"/>
7.     </layout>
8.   </head>
9.   <body>
10.    <par dur="5s">
11.      
12.      <text src="HelloWorld.txt" region="Text" />
13.      <audio src="HelloWorld.amr" />
14.    </par>
15.  </body>
16. </smil>
```

Listing 2.5: Exemplary MMS message

2.2.4 Java ME technology

Since different devices possess various capabilities, Java programming language is also available in different flavors called editions (cf. figure 2.7). Java Platform, Micro Edition (Java ME)²⁸ is an implementation of Java optimized for resource-limited devices. It has its own virtual machines and separate sets of class libraries. Basing on the concept of *configurations* and *profiles*, Java ME is available for heterogeneous devices such as mobile phones or PDAs and more powerful devices such as set-top boxes or digital televisions.

A *configuration* specifies the core classes and capabilities of Java Virtual Machine (VM) for a family of devices with similar networking facilities, processing power, and memory size. Two configurations were defined for Java ME: the Connected Device Configuration (CDC) and the Connected Limited Device Configuration (CLDC). The Connected Limited Device Configuration is a subset of the Connected Device Configuration aimed at resource-constrained mobile devices and comes with its own JVM called Kilo VM (KVM). The Connected Device Configuration was designed for more powerful

²⁷ MMS supports the following formats: JPEG, GIF, PNG, WBMP, SVG for images, H.263, MPEG-4 for video and MP3, MIDI, WAV, and AMR for audio [NoSv01, p. 105].

²⁸ Formerly Java 2 Platform Micro Edition (J2ME).

appliances and also contains its own Java Virtual Machine - CVM²⁹. Table 2.5 provides an overview of the Java ME configurations.

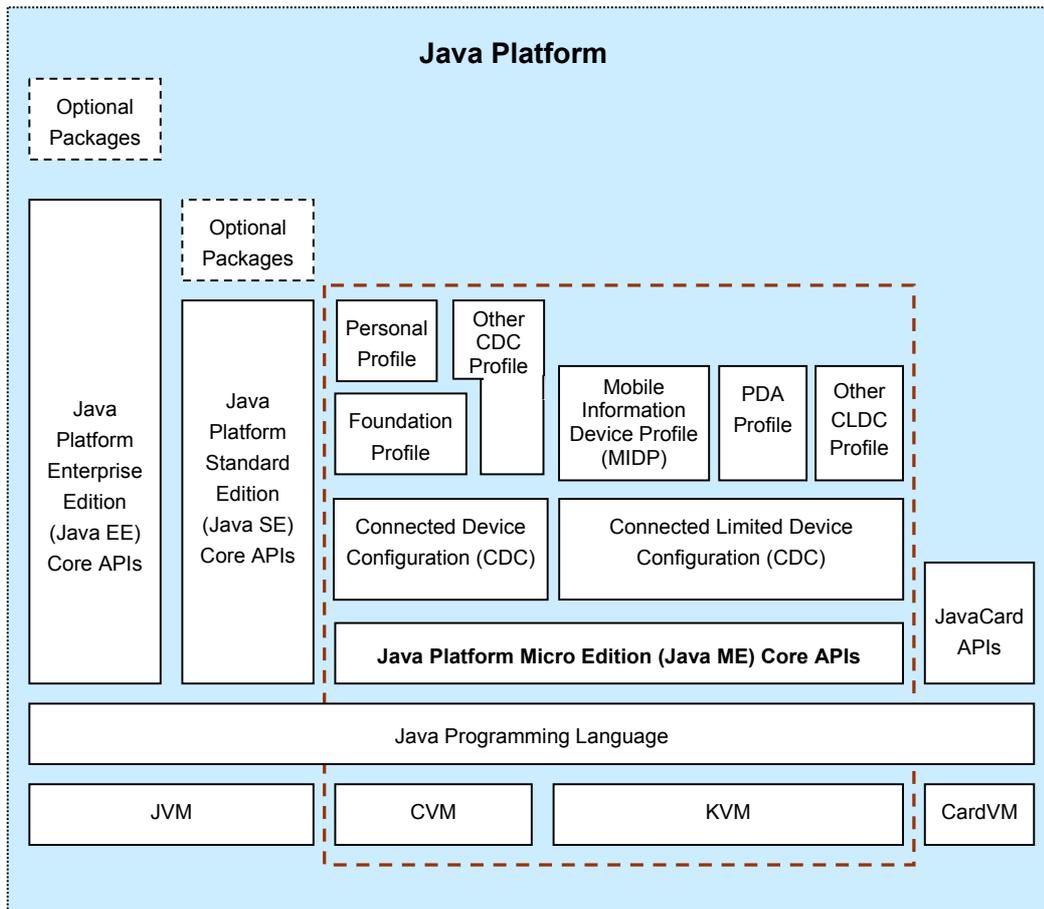


Figure 2.7: Environments within the Java Platform³⁰

A *profile* resides on top of a configuration. It contains a set of classes that have access to the underlying configuration classes and extends them. Mobile Information Device Profile (MIDP) is the most important profile for mobile devices. MIDP devices should possess at least the following features:

- a screen size of at least 95x54 pixels and of minimum 1 bit depth (black and white)
- at least one character input mechanism - one-handed keyboard (telephone keypad), two-handed keyboard (typical QWERTY keyboard) or a touch screen
- wireless networking
- memory capacity of 168 KB, including 128 KB memory for MIDP components, 32 KB for Java heap, and 8 KB memory for persistent data created in applications [Top102, pp. 48-50].

²⁹ At the beginning, CVM was an abbreviation for a Compact Virtual Machine. Since the KVM is also quite compact, this abbreviation is not used anymore [Top102, p. 228].

³⁰ Cf. <http://java.sun.com>.

Configuration	Typical Devices	Memory Requirements	Other features	Virtual machine
Connected Limited Device Configuration	mobile phones, PDAs, two-way pagers	160 KB (128 KB for ROM, 32 for RAM) to 512 KB available for the Java runtime	limited power, connectivity to some kind of network, often with a wireless, intermittent connection and with limited (9,6 Kbps or less) bandwidth	KVM
Connected Device Configuration	residential gateways, digital television sets, set-top boxes, organizers, high-end smartphones, and communicators	512 KB minimum ROM 256 KB minimum RAM	32 bit processor, connectivity to some kind of network, often with a wireless, intermittent connection and with limited (9,6 Kbps or less) bandwidth	CVM

Table 2.5: Overview of Java Platform Micro Edition configurations [Keog03, p. 12; Mahm01, pp. 3-4; Topl02, p. 11]

MIDP applications are called MIDlets and are similar to Java applets with limited capabilities. In contrast to applets, they do not have to be reloaded each time the user wants to view them, but remain on the device. Mobile Information Device Profile contains libraries for creating Graphical User Interfaces, basic network and storage, and a model for controlling applications. MIDlets can be applied to small-footprint devices (cell phones, two-way pagers), PDAs, and to devices with installed PalmOS version 3.5 and higher.

MIDlets are delivered to mobile devices in packages called MIDlet suites. A suite contains a JAR file with a manifest and a Java Application Descriptor (JAD) file. The JAD allows a device to determine whether it has the capabilities and resources needed by the suite. The JAR file can consist of one or many MIDlets, a manifest, and shared classes or resources. The JAR and the manifest share some attributes. This duplication results from the fact that JAR files are often quite large. Instead of downloading the whole JAR, a MIDP device first fetches its JAD file that can be transferred quickly. Basing on the information contained in the JAD, the user can decide about downloading the respective JAR file.

Common attributes included in the JAD are as follows: MIDlet-Name, MIDlet-Vendor, MIDlet-Version, MIDlet-Description, MIDlet-Icon, MIDlet-Info-URL, MIDlet-Data-Size, MIDlet-Jar-Size, and MIDlet-Jar-URL. In the manifest file, the name and the version of a MIDlet suite and class files corresponding to individual MIDlets are indicated [cf. Topl02, pp. 55-56]. Table 2.6 lists all the attributes, their possible locations (the manifest or JAD file), and provides a short description of each property.

All implementations of virtual machines should be able to load applications packaged in JARs. The device-dependent (i.e. designed and implemented by the manufacturer of a device) means of representing or accessing application code are, however, not included in the JVM [Topl02, p. 19]. For such tasks the so-called Java Application Manager (JAM) is responsible. It provides access to the Java environment for MIDlets and is in charge of installing, running, and removing MIDlets. Before running a MIDlet, its class file must be pre-verified. Pre-verification is necessary to provide faster and more simplified verification of class files once these files are loaded onto a mobile device. Class files that have not been pre-verified will fail to run [Mahm01, pp. 43-45].

Name	Description	Location	Optional
MIDlet-Name	Name of a MIDlet suite that identifies MIDlets to the user.	Descriptor (JAD) and manifest	No
MIDlet-Version	MIDlet suite version number. Can be used by the Java Application Manager (JAM) for controlling installations and updates.	Descriptor and manifest	No
MIDlet-Vendor	MIDlet suite vendor.	Descriptor and manifest	No
MIDlet-Icon	Name of the MIDlet's icon provided in a JAR file (in the PNG format). Can be used by the JAM to identify the suite in an application list.	Descriptor and manifest	No
MIDlet-Description	General description of a MIDlet suite. The user may use this to decide whether to install the suite or not.	Descriptor and manifest	Yes
MIDlet-Info-URL	A URL with more information describing a MIDlet suite.	Descriptor and manifest	Yes
MIDlet-Data-Size	Minimum persistent storage space needed by a MIDlet suite (in number of bytes).	Descriptor and manifest	Yes
MicroEdition-Configuration	Name and version of the minimum required Java ME Configuration (for example, ".CLDC-1.0").	Manifest	No
MicroEdition-Profile	Name and version of the minimum required Java ME Profile (for example, ".MIDP-1.0").	Manifest	No
MIDlet-<n> (one for each MIDlet)	Name, Icon, and Class for the n th MIDlet in a JAR, separated by commas. Name: Used to identify a MIDlet to the user. Icon: Name of an icon in the PNG format. May be left out. Class: Name of the base class for the n th MIDlet.	Manifest	No
MIDlet-Jar-URL	The complete URL pointing to a JAR file containing a MIDlet suite.	Descriptor	No
MIDlet-Jar-Size	The size of a JAR file (in bytes).	Descriptor	No
MIDlet-Delete-Confirm	A text message to be provided to the user when prompted to confirm deletion of a MIDlet suite.	Descriptor	Yes
MIDlet-Install-Notify	The URL to which a POST request is sent to confirm successful installation of a MIDlet suite.	Descriptor	Yes
<Any MIDlet specific attribute that does not begin with .MIDlet- ">	A MIDlet may add any number of custom attributes to the application descriptor as long as their names do not begin with the ".MIDlet" keyword. Values of these attributes can be retrieved at runtime.	Descriptor	Yes

Table 2.6: MIDlet packaging attributes [Top102, p. 54]

A MIDlet consists of at least one Java class that must be derived from the MIDP-defined abstract class `javax.microedition.midlet.MIDlet`. MIDlets run in an execution environment within the Java VM that provides a well-defined lifecycle controlled via methods of the `MIDlet` class. A MIDlet can also use methods from the `MIDlet` class to obtain services from its environment, and it must use the APIs defined in the MIDP specification, if it is to be device-portable.

MIDP contains high-level and low-level APIs for designing Graphical User Interfaces (GUIs). The purpose of the high-level API is to achieve portability across devices; it offers a limited control over the look and feel of an application. It is not possible to define the visual appearance (shape, color, or font) of high-level components. Most interactions with these components are already encapsulated in the implementation and applications are not aware of them. Figure 2.8 displays the most important classes used for the GUI design and table 2.7 provides a short description of high-level UI components and their categorization. Two types of UI components can be distinguished - `Displayable` components and `Item` components. The `Displayable` components encompass screens that can be displayed. The `Item` components are elements that can be aggregated within a screen [Piro02, p. 69]. The low-level API provides a complete control over the screen and access to the input devices. MIDlets that use the low-level API are not guaranteed to be portable, because this API provides device-dependent mechanisms [Top102, pp. 84-178].

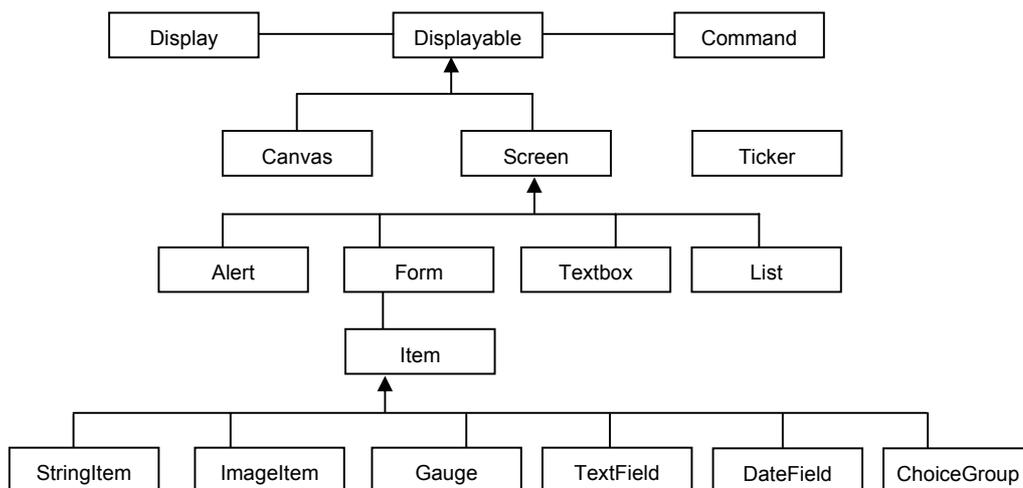


Figure 2.8: MIDP User Interface classes [Mahm01, p. 51]

MIDP offers the `javax.microedition.rms` package that helps to make data persistent. Additionally, the Generic Connection Framework (GFC) found in the `javax.microedition.io` package, defines a generic set of classes for Java ME I/O operations and network connectivity. Moreover, MIDlets support mobile media such as audio or video and, in version 2.0, provide useful functionalities for the development of mobile games. MIDP 2.0 expanded the connectivity possibilities beyond the HTTP standard supported in MIDP 1.0. HTTPS, datagram, sockets, server sockets, and serial port communication were included in MIDP 2.0 [Keog03, pp. 295-360].

MIDP UI Component Class Name	Description	Category of component
Alert	Informational pop up (modal/timed).	Displayable
ChoiceGroup	Group of selectable items, resides on a Form.	Item
Command	Semantic encapsulation of UI events.	-
DateField	Displays date and time.	Item
Display	Class that abstracts device display data structures.	-
Displayable	Ancestor of all components that can be displayed.	-
Font	Represents display fonts for text.	-
Form	Screen that aggregates items for display.	Displayable
Gauge	Visual gauge.	Item
Image	Representation of a PNG image.	-
ImageItem	Form resident representation of an image.	Item
List	List of selectable items.	Displayable
Screen	Abstract ancestor of all types of screens.	Displayable
StringItem	Form resident representation of a string.	Item
TextBox	Multiline, multicolumn text container.	Displayable
TextField	Single-line text container.	Item
Ticker	Representation of a ticker tape.	-

Table 2.7: MIDP UI components [Piro02, p. 69]

MIDP 2.0 also added a push functionality to the MIDlets. The `javax.microedition.io` package includes the `PushRegistry` and `PushListener` classes, and the `ServerSocketConnection` interface to interact with inbound socket and datagram connections. MIDlets may statically or dynamically register for the notification of incoming connections. Java ME applications are activated by the device application manager when it receives information from a pre-defined server. This push architecture makes it possible to leverage event-driven capabilities of devices and carrier networks. Alerts, warnings, messaging, and broadcasts can be easily provided and no specific network gateways (e.g. SMS gateway) have to be used [Orti03].

The Sun Java Wireless Toolkit (formerly under the name Java 2 Platform, Micro Edition (J2ME) Wireless Toolkit)³¹ is a toolbox for developing mobile applications based on Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP). The toolkit includes some emulators for viewing MIDlets and can also be used in conjunction with a third-party emulator (e.g. a Palmtop simulator).

The `javax.microedition.midlet.MIDlet` abstract class defines three lifecycle methods that are called during MIDlet's state transitions: `pauseApp()`, `startApp()`, and `destroyApp()`. When a MIDlet is first started, it is placed in the paused state. After this, the controlling software puts the MIDlet

³¹ <http://java.sun.com/products/sjwtoolkit/>.

in the active state (`public void startApp()` method) and the user can interact with it. The application can be placed back in the paused state by either the MIDP system or a programmer (using the `public void pauseApp()` method). The application can also be moved to the destroyed state from either the paused or active state (with the help of the `public void destroyApp(boolean unconditional)` method). In the destroyed state, the MIDlet should release all resources to the MIDP system [Koeg03, pp. 42-43].

In order to download a MIDlet from a server (the so-called Over The Air provisioning, OTA), the user has to fetch an HTML page with a link to a JAD. The request to retrieve the JAD is sent by the mobile browser and the response is passed to the application management software of the phone (JAM). Browsers are able to identify JAD files with the help of the MIME type “`text/vnd.sun.j2me.app-descriptor`”, sent by a server. After receiving the JAD, the application management software displays the content of the application descriptor and the user can decide whether to install the MIDlet suite. If the user wants to install the MIDlet, the application manager looks for the MIDlet-Jar-URL attribute in the JAD file and sends a request to that URL for the JAR. The server returns the JAR with the MIME type “`application/java-archive`” and the MIDlet is installed by the JAM [Topl02, pp. 77-78]. The whole process is depicted in figure 2.9.

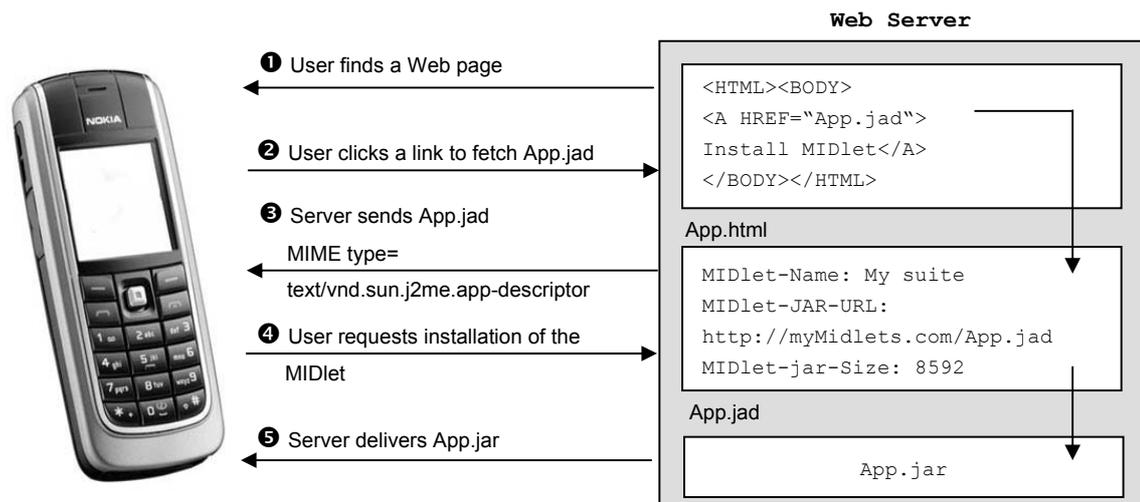


Figure 2.9: Over The Air (OTA) provisioning

Listing 2.6 provides the code of a very simple MIDlet. The most important part of this small MIDlet is the `startApp()` method. In this method, a display containing a text box with the text "Hello World" is created and the user sees this text box presented on the device. Other relevant methods are not implemented, but were declared.

```

1. import javax.microedition.midlet.*;
2. import javax.microedition.lcdui.*;
3. public class HelloMidlet extends MIDlet {
4. // The display for this MIDlet is created
5. private Display display;
6. // TextBox to display text
7. TextBox box = null;
8. public HelloMidlet() {}
9. public void startApp() {
10. display = Display.getDisplay(this);
11. box = new TextBox("Simple Example", "Hello World", 20, 0);
12. display.setCurrent(box);}
13. public void pauseApp() {}
14. public void destroyApp(boolean unconditional) {}}

```

Listing 2.6: Exemplary MIDlet code

Mobile devices can serve as thin or fat clients. In the first case, the devices are responsible only for presentation. Fat clients are additionally in charge of business logic. Examples of thin clients are mobile devices with browsers supporting WML or HTML. Fat clients are wireless devices with Java ME support. Table 2.8 outlines the most important differences between WML/XHTML-enabled devices and devices with Java ME/MIDP support.

Application categories	Characteristics	Examples	Support in WAP	Support in Java ME
Local applications	Applications can run on devices without network coverage	Local games, calculator, memo pad, etc.	NO	YES
Local storage	Client applications can store data persistently across usage sessions	Store current application state, store message for later sending if a network is unavailable	NO	YES
End-to-end connections	Direct connections between a device and another entity without the use of a gateway	Direct data connection to a corporate database	NO	YES
Call control	Perform operations triggered by incoming calls and setup voice calls	Redirecting calls to answering service, call a number in a message	YES	NO
Peer-to-peer networking	Direct device-to-device network applications without fixed-end server	Direct radio-to-radio chat, peer to-peer networked games, etc.	NO	YES
Very thin client/server applications	Device-side client has very limited application logic	Content browsing applications	YES	YES
Intelligent client/server applications (fat clients)	Device-side client has more application logic and local processing	Wireless IT applications, lower latency client/server application or games	NO	YES
Distributed applications	Applications run on multiple devices without a server	Multi-player games, server-less chat-room	NO	YES

Table 2.8: WAP standards (WML, XHTML) vs. Java ME [own research]

2.3 Types of mobile devices and their characteristics

During the past several years, mobile devices changed from communication devices to hand portable “Swiss army knives” equipped with primarily non-communicational features such as applications for data management, photo cameras, or music players. Many drawbacks of mobile handsets such as limited power or small displays remain in simple, mainly voice-based devices. Manufacturers of new generation phones try to provide new, optimal user experience by equipping them with larger displays, browsers supporting different markup languages (HTML, WML, and XHTML), or more powerful processors.

Current research goes one step further and aims at the development of “generic devices” that can be specialized for deployment but are not customized for any particular use. In 1999 Michael Dertouzos envisioned devices that are like oxygen: ubiquitous, essential, and mostly invisible, and started the Oxygen project [Dert99]. One of the goals of this project is to develop generic handheld devices, called H21s. They should provide mobile access points for users, accept speech and visual input, and reconfigure themselves to support multiple communication protocols. They should be able to perform a variety of functions and should serve as cellular phones, beepers, radios, television sets, cameras, or personal digital assistants. Recently, researchers from the Oxygen project demonstrated a new reconfigurable microchip. Mobile devices with this chip can change, chameleon-like, from a cell phone to a handheld computer and to a music player [cf. Scan04].

The term “mobile devices” describes devices that allow the user to complete computing tasks without being connected to a tethered network. In this thesis, the expressions mobile and wireless are used interchangeably. Different authors use diverse categorizations of mobile terminals. Weiss, for example, divides mobile devices into mobile phones, PDAs, and pagers [Weis02]. This classification is based on the UI conventions, functionality, and the primary use of devices. Handsets that combine all features of the aforementioned mobile devices are categorized as communicators. Prohm et al. use a different categorization and distinguish between basic phones, enhanced phones, smartphones, and wireless information devices [PrDiWo02]. In this thesis, mobile devices are divided into six categories [StMu02; Tara02]:

- 1) Web-enabled phones
- 2) low-end smartphones
- 3) high-end smartphones
- 4) Palm-sized PDAs
- 5) handheld PCs
- 6) tablet PCs.

Table 2.9 summarizes the most important features of different types of mobile devices. Examples of each type are provided in Appendix 1. In the taxonomy used, devices sometimes categorized as mobile such as laptops or task devices (e.g. bar code scanners) are omitted. The devices are classified according to the characteristics that may influence the design of mobile solutions. The most important features relate to input and output interactions with wireless devices and encompass:

- browsers and supported markup languages (alternatively, support for Java ME)
- operating systems
- memory and processing power
- interaction style/input capabilities (keyboard, touch screen, voice, etc.)
- output capabilities (display, voice, etc.)
- supported network and data connection speeds.

Device type	Examples	Characteristics
Web-enabled phones	Nokia 6510	Screen: limited display, 4-12 lines of text (10-20 characters per line), monochrome or color Data input: 12-button phone keypad, voice input Operating system: none Memory: 2-128 MB (small to medium memory) Processing power: 90-150 MIPS DSPs (low processing power) Markup/programming languages: not supported, eventually WML
Low-end smartphones	Nokia 5100 Nokia 7600	Similar to Web-enabled phones, but equipped with Java ME Better processors, larger memory and storage facilities
Palm-sized PDAs	Palm Tungsten T5, Palm Zire 72	Typical screen: 160x160, 320x240 VGA, supporting colors Data input: touch-screen capabilities for user input, stylus, QWERTY keyboard, voice input Memory: 8-512 MB, usually 256 MB, additional MMC possible Processor speed: 33-624 MHz Operating system: Microsoft Pocket PC, Palm OS, Microsoft Windows Mobile SE/Premium Markup/programming languages: HTML, XHTML, Java ME, proprietary formats
High-end smartphones	Nokia 6670, Nokia 7710, Sony Ericsson P800 Smartphone	Typical screen: 640x200 to 320x240 Data input: keyboard, touch screen Memory: typically 64 MB RAM and 64 MB Flash memory; up to 512 MB with MMC possible Processor speed: about 400 MHz Operating system: Palm OS, Symbian OS, Microsoft Pocket PC Phone Edition, Microsoft Smartphone Markup/programming languages: WML, XHTML, HTML, Java ME
Handheld PCs	Samsung NEXiO Handheld PC	Typical screen: 480x320 VGA Data input: QWERTY keyboard, touch screen Memory: 256 MB to 5 GB (internal flash drive, additional program memory for applications and data) Processor speed: 416 MHz and higher Operating system: Palm OS, Microsoft Pocket PC, Microsoft Handheld PC Markup/programming languages: HTML, XHTML
Tablet PCs	Acer TravelMate100 Tablet PC	Typical screen: 10.4"-14.1" TFT XGA Data input: pen-based input, touch-screen, QWERTY keyboard Memory: 256 up to 2 GB SDRAM, 20 GB to 80 GB hard drives Processor speed: from about 700 MHz up to 1.1 GHz Operating system: Microsoft® Windows® XP Tablet PC Edition Markup/programming languages: HTML, XHTML

Table 2.9: Types of mobile devices [general classification based on SmBrKr01; Tara02; own research]

In comparison to traditional desktop computers, mobile devices follow a different usage paradigm and possess certain limitations. They have a comparably lower computational power, smaller memory and cache as well as limited storage capabilities. Currently, many mobile devices use a secondary type of memory that can significantly increase the available capacity. It is inserted into the device as a slide-in card. Various types of storage media are available, for example, Memory Stick, Secure Digital (SD), and Multimedia Memory Card (MMC).

Since the performance of a processor is directly related to the power it uses, mobile devices do not have high processor speeds in order to save their batteries. The cost factor is another reason for applying slow processors in wireless devices – fast processors would be quite costly and mobile devices are expected to be cheap to attract customers.

The following user input components can be implemented to facilitate input [cf. Keto02; Kilj04]:

- numeric keypad or different types of keyboards
- control keys and devices for controlling the device (navigation keys, joysticks, rocker keys, rollers, wheels, softkeys, menu keys, backstepping keys, etc.)
- call-management keys for managing phone calls
- volume keys for quick access to control audio volume
- power key to switch the device on/off
- special-purpose keys to access dedicated functionality, i.e. camera, Internet access, etc.
- microphone for audio input
- digital camera
- sensors (light, fingerprint recognition sensor)
- touchpad or touch screen for direct manipulation, UI control, or writing recognition.

Furthermore, the following output devices can convey information to the user [cf. Keto02; Kilj04]:

- flat-panel display or displays
- LED(s)³² to indicate the status of the device (e.g. low battery)
- earpiece, hands-free loudspeaker for audio output
- buzzer for playing ringing tones and other audio
- vibration motor for tactile output
- laser pointer, or flashlight.

Screens of mobile devices vary in display dimensions, supported fonts and text sizes, and the number of displayed colors. A reasonable screen size for Web site design is 320 x 240 pixels (known as one Quarter Video Graphics Array or QVGA). A portrait QVGA screen (240 pixels across by 320 pixels high) can display around 16 rows of 24 characters of text. The size of a handheld screen has similar width to a newspaper's column that can display 22 lines of text, 30 characters wide. Some of the Palm-sized PDAs of a new generation possess the Video Graphics Array (VGA) screen size (640 x 480 pixels) but typical mobile phones usually come with a screen of 176x208.

³² A light emitting diode (LED) is a special diode with a semiconductor chip that emits incoherent narrow-spectrum light.

In the future, flexible screens may allow displays which will be rolled or folded up. E Ink and Gyricon Media³³ are developing displays with electronic ink technology, currently only in black and white. Such screens hold an image until voltage is applied and they use less power than traditional LCD displays [Tara02]. Even though the screen resolution of mobile devices is continuously increasing, their physical size will remain limited to the size of a pocket.

Due to restricted input methods, text input is much slower on mobile devices than with the help of a PC keyboard. Some devices provide support only for vertical scrolling and it is not possible to activate GUI components using a pointing device. Input facilities are device-dependent. Most mobile phones use a numeric keypad consisting of 12 keys (0-9, *, #) for text input. Text entry on a numeric keyboard is accomplished using the so-called multitap technique - the user presses a key repeatedly to cycle through the letters displayed on this key. The characters "abc" traditionally appear on the second key; pressing this key once yields "a", twice "b", etc. One of the most popular techniques for text input is the predictive text input software T9³⁴. T9 computes all possible combinations of a sequence of key presses, looks them up in a dictionary and displays them on the screen. A fasttap is a multi-level keyboard that separates the letters and numbers what increases the speed of text input (cf. figure 2.10).

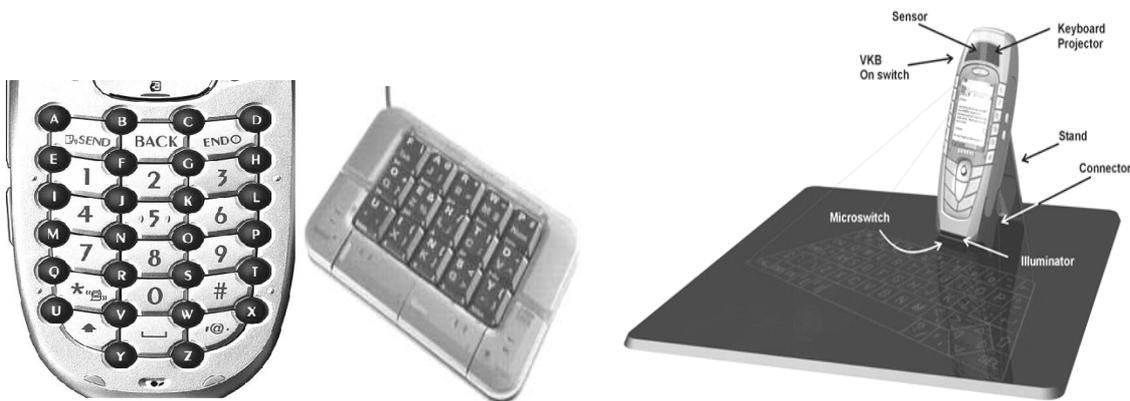


Figure 2.10: Mobile keyboards: a) fasttap, b) frogpad, c) VKB virtual keyboard

Two-handed devices can have a standard QWERTY keyboard, a thumbboard, a frogpad, or more advanced solutions such as virtual keyboards (cf. figure 2.10). Thumbboards were popularized by RIM Blackberry devices. They are two to three inches wide and can be operated with two thumbs. In frogpads, the most frequently used letters are accessible with a single press. A modifier key gives the user access to the remaining letters using simultaneous key presses. Virtual keyboards are a very interesting solution. Instead of using a physical keyboard, the user types in the air. In VKB virtual keyboards³⁵, a laser draws a keyboard onto a nearby surface and detects which lines are broken to

³³ Cf. <http://www.eink.com> and <http://www.gyriconmedia.com>.

³⁴ Cf. <http://www.tegic.com>.

³⁵ Cf. <http://www.vkb-tech.com/>.

determine which keys were activated. In Senseboard virtual keyboards³⁶, straps with sensors detect finger motions and map these motions to text [cf. WiBa04].

PDAs and Palmtops usually possess a touch screen with a soft keyboard displayed on it and a pointing device to write directly on the screen. Technologies such as character, word, or handwriting recognition help to enter words faster [cf. BeRiSt03, p. 27]. The character and handwriting recognition are demonstrated in figure 2.11. As an additional input mechanism, voice-based data input may be implemented (e.g. using VoiceXML and an appropriate ASR engine).

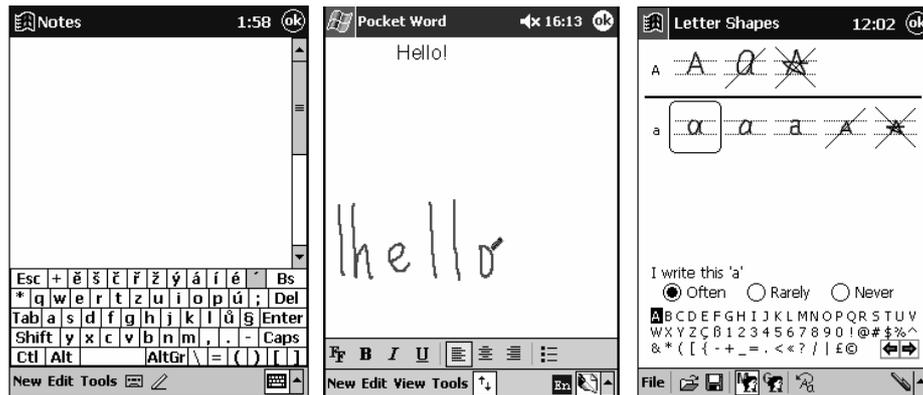


Figure 2.11: Text input possibilities: a) soft keyboard, b) character recognition, c) handwriting recognition

Operating systems provided for mobile devices offer a functionality which is limited in scope (e.g. no support for threads or processes for background tasks). The most popular operating systems for mobile devices are Symbian OS, Microsoft Pocket PC (formerly Windows CE), and Palm OS. Linux-based systems may also be an alternative in the future.

Wireless devices may support different markup languages or possess the ability to load and install Java ME applications. Most of the old generation phones are equipped with a browser interpreting WML. New generation phones come with a built-in support for XHTML MP and/or HTML.

It is estimated that the average display size increases by 5 percent in one year and the processing speed and memory double every 18 months [cf. Funk03, p. 29]. The short-term effects of these developments are improved user interfaces and better download speeds, the long-term effect can be the merger of different device types and development of generic devices.

³⁶ Cf. <http://www.senseboard.com/>.

3

Mobile content and its adoption

In the 1950s Harold Osborne, a chief engineer at AT&T, foresaw that mobile telephony would enable users ubiquitous access to information via portable devices. He predicted that people would be able to communicate using small devices and that they would see each other in three dimensions [Ling04, p. 4]. Even though the Osborne's vision became true, the acceptance of mobile solutions is still below the level projected by experts. They anticipated a new boom, similar to the previous Internet and e-commerce booms, resulting from the growing convergence of mobile telecommunications and the Internet and the increasing number of added-value services for mobile devices. In comparison to the traditional fixed Internet, mobile Internet enables users access to content from anywhere and at any time. In mobile environments, however, users have to cope with various limitations of mobile devices that are not typical for desktop computers. Small displays, unstable connections, or short battery lives are only some examples of such limitations. Since many relevant factors, unique to the mobile context (e.g. mobile interface design principles or the development of appropriate pricing models for mobile content), are not taken into account in mobile services, the adoption of mobile Internet in some countries is still below expectations.

This chapter focuses on the past developments in the area of mobile solutions and possible future directions in this domain. Section 3.1 presents the definitions of mobile Internet, mobile commerce, and mobile business and outlines areas in which mobile applications can be applied. Section 3.2 describes the market for mobile devices and applications. Section 3.3 provides an overview of the adoption of wireless applications worldwide and briefly describes the reasons for the varying acceptance of mobile solutions.

3.1 Mobile services and their application domains

Mobile applications and services are usually summarized under the definitions of mobile commerce, mobile business, or mobile Internet. Authors from the industry and researchers use various explanations for these three terms. For example, Lehman Brothers apply a quite broad definition and characterize m-commerce as "the use of mobile handheld devices to communicate, inform, transact, and entertain using text and data via connection to public and private networks" [SkJoDi00, p. 8]. Similarly, Elliot and Phillips state that mobile commerce is concerned with the use, application, and integration of wireless telecommunication technology and wireless devices within the business systems domain [EIPh04]. According to these descriptions, messaging or information services offered at no cost also fall into the m-commerce category.

Since the Compact Oxford English Dictionary specifies the term “commerce” as “the activity of buying and selling, especially on a large scale”³⁷, many authors put particular focus on the monetary aspects of m-commerce. Durlacher Research defines m-commerce as “any transaction with a monetary value that is conducted via a mobile telecommunications network” [Muel00]. In this case, SMS messages from one person to another do not belong to the m-commerce category, while paid SMS messages from an information service provider to a consumer are regarded as being m-commerce. In this thesis, mobile commerce is considered to be an extension of e-commerce and is therefore defined as “the ability to buy and sell products and services with the help of mobile devices. It includes online display of goods and services, ordering, billing, customer service, and all handling of payments and transactions.”³⁸

Mobile commerce encompasses four basic categories: mobile entertainment, mobile communication, mobile information, and mobile transactions [SkJoDi00]. To the mobile entertainment category belong services such as music, games, and video. Mobile transactions include among others mobile banking and brokerage, shopping, and booking. Mobile communication consists of services like e-mail and SMS. Examples of mobile information services include news, weather, city guides, and stock market information.

The availability and quality of mobile services usually relates to the capabilities and prices of mobile networks. Consumers of mobile commerce services are individual customers, employees, and/or business partners (including government organizations). Therefore, B2B (business-to-business), B2C (business-to-consumer), B2E (business-to-employee), B2A (business-to-administration), and P2P (peer-to-peer) can be distinguished as mobile commerce categories [Dete04].

Mobile commerce is often confused with mobile business. According to some authors, mobile business focuses on the monetary exchange of products and services through wireless devices [Zobe01]. Alternative definitions concentrate on the mobile access to business-relevant information systems or data, including internal systems of a company [TeC03]. Similarly to the concepts of e-commerce and e-business, some researchers adopt a broader view of mobile business that includes “all activities related to a *potential* commercial transaction through communications networks that interface with mobile devices” [Lehn03; Tara02]. Extending the definition of electronic business, mobile business is defined in this thesis as “the process of using mobile technology to help businesses streamline processes, improve productivity and increase efficiencies. It enables companies to easily communicate with partners, vendors and customers, connect back-end data systems and transact commerce in a secure manner.”³⁹ Mobile commerce is therefore treated as part of mobile business. Figure 3.1 shows the main services and actors in the mobile business and mobile commerce environment.

³⁷ Cf. http://www.askoxford.com/concise_oed/commerce?view=uk.

³⁸ <http://www-5.ibm.com/e-business/uk/glossary>.

³⁹ <http://www-5.ibm.com/e-business/uk/glossary>.

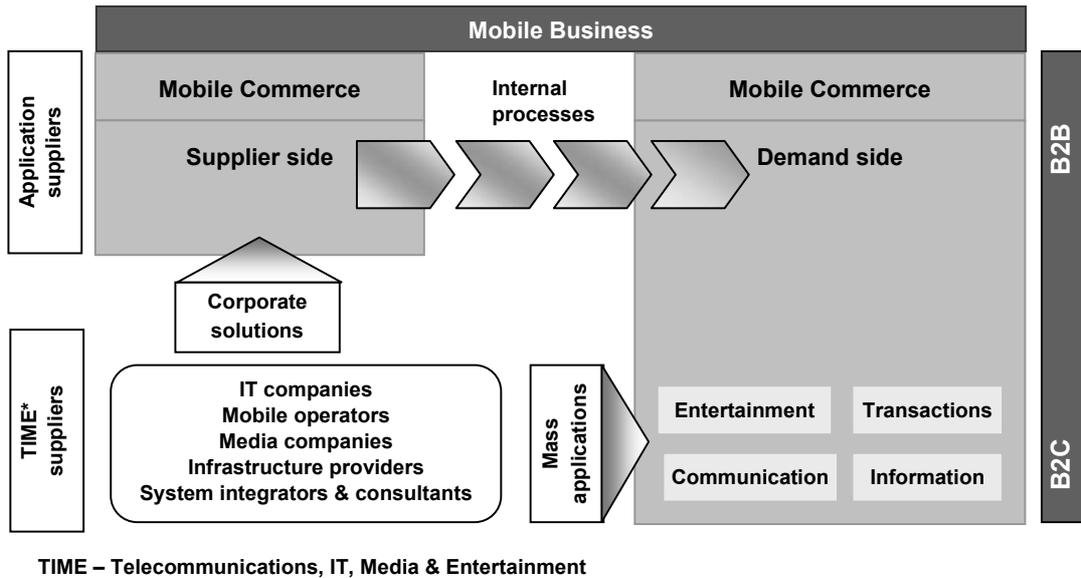


Figure 3.1: Mobile business and mobile commerce [Dete04]

Mobile Internet is the next term that appears in the literature in the mobile context. It is usually understood as an extension of the traditional Internet and is defined as the use of mobile devices to perform data transactions or to generate data traffic (e.g. e-mail, SMS, downloads) via a connection to mobile networks [AhBa02]. Mobile Internet is therefore a broader term than mobile commerce and does not necessarily include commercial aspects. All services and applications that are presented to the end-user over mobile Internet are defined as content. Under content one can summarize, for example, sites with information generated from databases or entered manually by content authors, text and pictures sent as messages, advertisements, e-mails, or games.

Mobile Internet is sometimes taken literally as the “wireless access to the digitized contents of the Internet via mobile devices” [ChKi03, p. 240]; the same services accessible in the traditional, fixed Internet are made available for mobile access. In an alternative view, mobile content is perceived as largely independent of the Internet context. According to this perspective, mobile services have to be designed in a different way because they are developed for devices with restricted capacities, limited bandwidth, and different pricing models for mobile networks [Tara02a].

Mobile services

Mobile services followed the evolution path of wireless networks (cf. figure 3.2): the more powerful the networks were, the more advanced services could be offered. Simple communication (voice and text messaging) and information were the most frequently used services in second generation networks. 2.5G and 3G services put focus on advanced personal communication, rich mobile content, and mobile media. In the area of personal communication, 3G enables sending messages which contain text, pictures, and animations; it also makes chatting and live video conversations possible. It is assumed that with the proliferation of 3G networks, particularly UMTS, users will want to access personal information (e.g. calendars) or corporate information, and will also benefit from mobile information services such as news, stock exchange information, or local guides [UMTS03]. Due to the improved capabilities of modern mobile devices, online services can further be enhanced by multimedia content

such as music, animated pictures, rich voice, and video telephony. To this category belong services such as conferencing, telemedicine, teleworking, and multimedia communication. The role of mobile phones has radically changed in the past few years: they became portable entertainment players, small digital cameras, marketing tools for retailers and manufacturers (e.g. SMS-based marketing), shopping devices and navigation tools, new types of tickets and money, and new mobile Intra-/Internet devices.

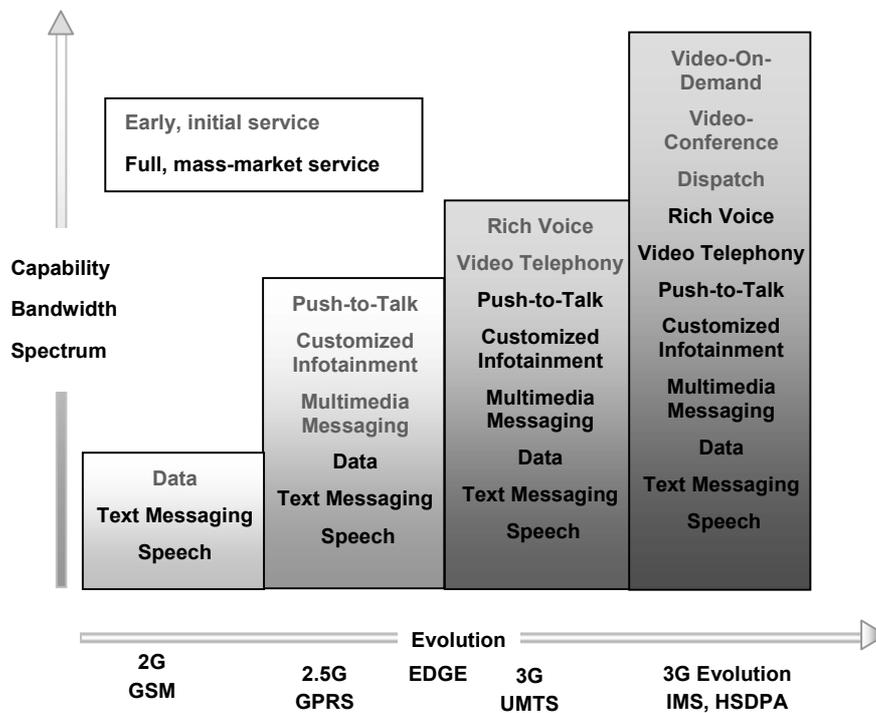


Figure 3.2: Development of mobile services [UMTS03, p.11]

Mobile services are generally designed for two segments of users: private customers and business users. Mobile content providers consider services tailored to the needs of corporate users as their future sources of revenues. Different studies predict that mobile business will blossom and users will benefit from mobile solutions in different contexts. Examples of potentially successful services include [cf. LeKuLe04]:

- communication-oriented services, mobile entertainment (i. e., sex and gambling)
- mobile workforce tasks such as order taking or stock inquiries at customers' locality, job-dispatch and scheduling to traveling/mobile personnel
- access to core enterprise information systems from anywhere at any time
- reservation of tickets
- access to news, stock market data, and information services
- mobile marketing, banking, and brokerage
- mobile payment
- mobile supply chain management
- transport fleet management
- location-based services.

Wireless technologies are supposed to bring enterprises considerable gains in productivity and efficiency [cf. GeSh04; NaSiSh05]. Mobile solutions can enhance, for example, efficiency by distributing information to the mobile workforce and by offering new channels of interaction with customers [LeAn01]. Further benefits include lowered operational cost, faster access to critical data, better asset usage, increased customer satisfaction, and improved decision-making [Korn+04; Vars+02]. Table 3.1 lists some typical mobile application categories and some enterprises, in which they have already been applied. Table 3.2 provides an overview of applications that are seen as particularly promising for the future use.

Category	Application	Selected examples
Communications	E-mail, messaging, video conferencing Personal Information Management	Lufthansa Systems, Britannia Airways IBM Global Services
Supply Chain Management (SCM)	Asset Tracking Inventory Management	Ford, Pepsi, Hertz, UPS
Customer Relationship Management (CRM)	Sales Management & Field Service Contact Management	Merck, Nextel, BellSouth, Bayer
Enterprise Resource Planning (ERP)	Asset and Order Management Invoice Generation	Shell, Chevron Texaco, US Navy, Xerox
Human Resources (HR)	Time and Expense Tracking	Allstate Insurance
Mobile Commerce	Mobile Banking Alerts	Bank of America, E-Trade, Intercontinental Hotels, Delta Airlines

Table 3.1: Mobile enterprise applications [Baso05]

	Business	Customer	Employee	Administration
Business	B2B <i>Sales & Supply</i>	B2C <i>Payments</i>	B2E <i>Marketing & Sales</i>	
Customer	C2B <i>Ticketing & Reservation</i>			C2A <i>Mobile Government</i>
Employee	E2B <i>Mobile Workforce</i>		E2E <i>Team Management</i>	
Administration				

Table 3.2: Application trends in mobile business⁴⁰ [based on LeKuLe04]

Mobile enterprise applications can be advantageous for employees, customers, suppliers, and business partners. Supply Chain Management (SCM), Enterprise Resource Planning (ERP), and electronic marketplaces are the most popular application domains for mobile B2B. Among employees

⁴⁰ Darker shading indicates that this application field is expected to be particularly successful.

who can benefit from mobile devices are sales force and field service workers, executives, managers, and office workers. They usually need to obtain access to core business information systems in order to perform their work-related activities. Table 3.3 presents some common application domains of mobile solutions for employees and indicates their main benefits.

Opportunities	Benefits
Communications: <ul style="list-style-type: none"> - Basic e-mail - SMS text messaging - Alerts and notifications 	Productivity: <ul style="list-style-type: none"> - Improved employee productivity - Improved sales force productivity - Improved field force productivity
Personal Information Management (PIM): <ul style="list-style-type: none"> - Calendar, contacts, tasks - To-do lists, memos 	Delivery of time-sensitive and/or location-relevant information
Intranet access: <ul style="list-style-type: none"> - Company directory, office locations - Employee directory - Travel arrangements, reservations - Time and expenses reporting 	Cost reduction
Internet access: <ul style="list-style-type: none"> - Company Web site and applications - Competitive intelligence 	Reduced asset downtime
Sales force applications: <ul style="list-style-type: none"> - Customer/account information - Product/service/pricing information - Order entry and quoting - Inventory management - Competitive information, sales reporting 	Reduced resource costs (such as phone, fax, printing, mailing)
Field force applications: <ul style="list-style-type: none"> - Dispatch, project lists, proposals - Service histories - Inspection forms - Product and part information, order processing - Time and expense reporting 	Revenue generation, increased sales
Enterprise Resource Planning applications	Better knowledge/decision making
Enterprise dashboard / Business Intelligence applications: <ul style="list-style-type: none"> - Key performance indicators (KPIs) - Alerts and notifications - Financial and operational monitoring - Reporting 	Improved executive reporting and decision making
Improved data capture and accuracy	Delivery of time-sensitive and/or location-relevant information Increased satisfaction of employees and customers, better customer service, increased competitive advantage

Table 3.3: Opportunities and benefits of mobile applications for employees [Evan02]

In a study conducted as part of the Mobile Internet Business (MIB) project, over 300 case studies were analyzed in order to find the main benefits of the deployment of mobile solutions across different industries [KuKr06]. The authors state that the application of mobile solutions brought the following advantages to the enterprises [KuKr06, pp. 54-55]:

- improved information flows - 27 percent, with particular focus on the following aspects:
 - gathering of information independently of the location - 35 percent
 - possibility to obtain up-to-date information - 26 percent
 - elimination of disadvantages resulting from the changes in the devices used during the process of gathering or processing of information - 23 percent
 - benefits resulting from the information entered on client's site - 16 percent
- faster workflows and processes - 18 percent
- improved work with customers - 13 percent
- structural improvements with regard to processes - 11 percent
- quality improvements - 7 percent
- cost decreases - 7 percent
- improved flexibility - 5 percent
- better process transparency/visibility - 5 percent
- technological benefits - 3 percent
- positive motivation effects - 1 percent
- increased turnover/profit - 1 percent
- other - 1 percent

Although most managers are convinced that wireless technology can be of great importance for their companies, they usually have no idea how to apply this technology successfully [LeKuLe04]. They see great potential in business-to-employee (B2E) applications (e.g. solutions offering access to internal enterprise information systems) and logistics but are not able to elaborate any clear business strategy.

The general idea of content providers and mobile operators is to seek for the so-called "killer applications" that will immediately lead to great success and mass market. Various types of combinations were proposed [cf. CaWa02; CaWa02a]:

- *Killer Cocktail* - individual components cannot be distinguished in the mix
- *Killer Pizza* - components can be distinguished in the mix
- *Killer Bouquet* - the aggregate of all components is greater than the sum of its parts (e.g. mobile transactions combined with mobile payment)
- *Killer Soup* - the quality improves with the number of components, but an operator is needed to stir (e.g. i-mode)
- *Killer Fondue* - similar to the soup, but an operator is not necessary.

However, the "killer application" approach seems to be wrong due to its focus on technological aspects. Following this approach, new applications are expected to conquer the market only because they offer something that was previously not technically feasible. Companies should rather look for "killer user experiences" and develop applications based on user needs instead of technological advances and possibilities [cf. EdKo04]. For example, the mobile industry is currently investing heavily

in the development of the so-called rich media applications (videos, audio, and pictures). It particularly focuses on video-based communication in which the users can see each other and on mobile television. From the user's point of view, looking at the communication partner or watching television in miniature may not be of real use⁴¹. He/she may be rather interested in checking the current location of his/her children or obtaining warnings about natural catastrophes.

3.2 Market for mobile devices and mobile applications

Within ten years, mobile telephony changed its status from a technology used by a narrow group of customers to a mainstream technology, used even by children. In 2003, mobile phone subscriptions had overtaken fixed telephone subscriptions for the first time, as shown in figure 3.3 [Cast+04, p. 5; UMTS03]. The distribution of mobile penetration rates differs worldwide: the highest rates are noticed in Europe, followed by Oceania and North America. The International Telecommunication Union (ITU) estimated that in 2004 the penetration rates were as follows: Africa – 9 percent, Asia – 34.3 percent, both Americas – 42.6 percent, Europe – 71.4 percent and Oceania – 62.8 percent [ITU05, p. 6]. Although the U.S. has one of the most competitive wireless communication systems, wireless communication technology in this country still exhibits lower acceptance than in other developed countries. This is partially due to the facts that local calls in this country are free and several incompatible network standards are available. For a long time, American phone owners were not even able to send SMS messages to people using a different network.

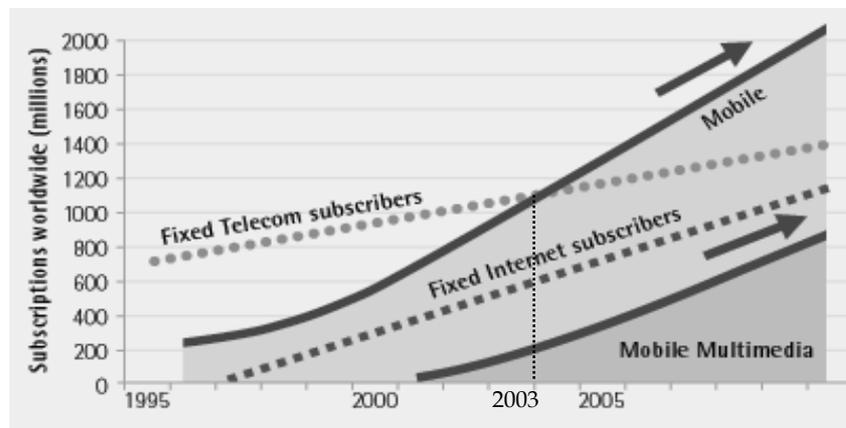


Figure 3.3: Mobile, Internet, and fixed telephony subscriptions [UMTS03, p. 2]

Strategy Analytics estimated that wireless subscriptions approached 2.2 billion at the end of 2005 (with 1.8 billion unique users). It is expected that they will pass 3.5 billion by 2010, but more than 70 percent of users will be on prepaid plans. 2005 was also the breakthrough year for 3G services: the number of W-CDMA users grew to 49 million and CDMA2000 1x EV-DO users reached 26 million. Analysts predict that these figures will double in 2006, rising to 101 million and 53 million, respectively.

⁴¹ Surprisingly, the results of different studies on mobile TV show unexpected interest in such applications. For example, in a recent Siemens study (March 2006) fifty-nine percent said they would like to watch television on handheld devices. http://news.yahoo.com/s/afp/20060308/tc_afp/afplifestylegermany.

However, the average service revenues fell despite the growth in the number of subscribers. Voice-based services are still popular: in 2005 voice traffic increased by 30 percent to 5.7 trillion minutes. Due to high saturation of mobile phones market, the penetration rates will begin to flatten in Europe and North America. The highest growth rates are expected in Asia, Africa, and Latin America. Global wireless service revenue is expected to rise 11 percent to \$623.9 billion, according to the Strategy Analytics report. The mobile industry is expected to generate \$800 billion in revenue in 2010. Emerging markets will probably account for about 42 percent of the total. New features and services such as music and video downloads are expected to compensate the decrease in revenue in established markets [cf. Kend06].

The biggest mobile manufacturers are Nokia (54 percent of the market in 2005), Motorola, Palm, and Research in Motion (RIM). Handsets are becoming more and more sophisticated. Built-in cameras, wireless connectivity through Bluetooth, support for MMS, high-speed data access, and downloadable applications are going to be a standard in 2006 [Ambr06; Litt04]. The number of smart and enhanced phones overtook the number of PCs already in 2003. The growth will continue, but the ratio of PDAs and smartphones will increase, whereas the number of simple “basic phones” will decrease [Dete04, p. 62]. Handset sales grew 21 percent in 2005 to 823 million units worldwide. They are expected to reach 1 billion units in 2007. In established markets, replacement sales are slowing (with the annual growth of 4 percent in Europe and 3 percent in North America in 2005). Double digit sales growth is expected in emerging markets in Asia Pacific (India), Central and Latin America (Brazil, Mexico), and Central and Eastern Europe (Russia) [Ambr06].

Market for mobile applications

The market for mobile applications is growing slowly. It is expected that voice will remain the basic source of mobile revenue and messaging will mainly be responsible for the non-voice revenue growth. The influence of text messaging is very strong. As a meaningful example, the findings of the British Medical Journal can be mentioned. Smoking was down for teenage girls in the UK, because they spent money on mobile phones instead of cigarettes [Jens03, p. 9]. Analysts expect that text messaging will continue to outshine its more complicated multimedia relatives (MMS). Picture messaging may increase due to two factors: the proliferation of high-quality camera phones and expected lower prices for such services. Mobile e-mail will grow in personal and enterprise markets [Litt04].

However, traditional messaging, the main source of revenue for mobile operators, may be endangered by new initiatives. For example, Hotxt, the British start-up company, launched in April 2006 a new Internet-based messaging service [Econ06]. The users have to download the Hotxt application, log into the service with a user name and can then exchange an unlimited amount of messages with other Hotxt users for only £1 per week. Instead of using the conventional infrastructure with SMS centers and gateways, messages are sent as data packets across the Internet and the user pays only for data

transport. Even if Hottxt texting will not eliminate traditional SMS messaging, it can significantly cut the margins on text messages⁴².

Mobile Internet and the e-mail use have grown in Western Europe and America. 56 percent of all multimedia phone owners from 21 countries interviewed in the Mobinet study declared that they have already browsed the portal of their operator (e.g. Vodafone *live!*) or used the mobile e-mail at least once a month. Consumers use mobile media services more intensively than in previous years - they send pictures, photos, or video clips. MMS has the highest penetration rate among 19- to 24-years olds, where almost half of multimedia phone owners confirm that they use the service regularly [AtKe+05].

According to the Mobinet study, entertainment services are not as popular as communication services [AtKe+05]. Mostly younger customers are willing to download music or play games. In 2005, 33 percent of consumers with multimedia devices declared that they downloaded music, up from 21 percent in 2004. Gaming has grown in Japan, the Americas, and Scandinavia. Users also show interest in new innovative solutions such as mobile TV. When asked about categories of mobile content, two-thirds of them expressed a desire for time-sensitive TV content (news, sport events) rather than entertainment programs (e.g. films).

Mobile services	UK		Germany	
	3G	2G	3G	2G
Sent text message	89.0%	83.0%	84.1%	75.9%
Used instant messaging	6.0%	2.8%	3.7%	2.4%
Downloaded game	10.2%	3.5%	7.0%	1.8%
Purchased ringtone	12.6%	6.6%	14.7%	6.7%
Captured video	47.0%	17.1%	30.8%	9.2%
Sent video to peer	18.9%	6.9%	9.8%	3.2%
Viewed short video clip	12.8%	1.4%	7.9%	0.6%

Table 3.4: Relative monthly consumption of mobile data services as of December 2005 [Metr06]

Recent findings of the first European benchmark survey conducted by M:Metrics show that 3G users in the U.K. and Germany increasingly exploit the multimedia capabilities of their handsets (c.f. table 3.4) [Metr06]. Capturing and sending videos over 3G networks is becoming more and more popular. Messaging services, gaming, and personalizing phones by downloading new content (e.g. ring tones) are also well-liked.

Figure 3.4 shows the projected revenue shares in 2008 per service category. The potential sources of future ARPU⁴³ growth are expected to be the entertainment (including personalization of phones) and

⁴² One SMS typically costs around 10p in the UK.

⁴³ The average revenue per user (ARPU) measures the average monthly revenue generated for each customer unit, such as a cellular phone or a PDA that a carrier has in operation.

media services as well as business data services [AtKe+05; Litt04; Metr05; Metr06]. Compared with the current situation, revenues generated by communication services are likely to decrease.

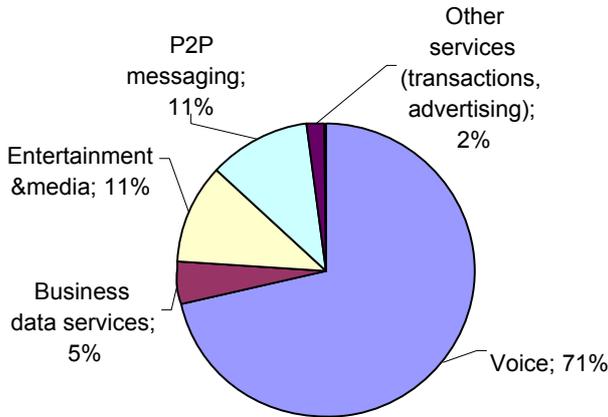


Figure 3.4: Expected global mobile service revenue in 2008 [Noki04]

Mobile applications are increasingly being used by companies. The estimated penetration rate of mobile phones in the working population in 2004 was about 27 percent [Litt04]. Gartner's annual survey of 1400 European Chief Information Officers (CIOs) revealed that mobile technology is one of their three priorities in 2005. The CIOs wanted to put a particular focus on spendings for mobile workforce [Gart05]. This priority seems to be justified: according to the IDC's study the mobile worker population will grow from over 650 million worldwide in 2004 to 850 million in 2009 [Drak+05]. This growth will further be supported by the adaptation of mobile solutions. Other statistics suggest that about half of mid-size and large enterprises use wireless data today and another 30 percent are planning or evaluating the use of mobile technology. Commonly applied mobile solutions include e-mail, mobile Internet, spreadsheets, and word processing documents [RysR05].

Table 3.5 shows the percentage of companies in which mobile workers have access to wireless applications. According to the research conducted by Nokia in 2003, an average of 40 percent of workers spent more than eight hours a week away from their offices. During this time, 29 percent of them could read mobile email and 7 percent could access company information systems and update the information stored in them [Noki03].

	Company size (number of employees)				
	1-9	10-99	100-249	250-499	500+
E-mail	31%	24%	27%	32%	36%
Internet	24%	25%	29%	30%	33%
Company systems	5%	7%	8%	15%	15%

Table 3.5: Percentage of companies where mobile workers have access to mobile data applications [Noki03]

A study conducted in Germany in 2004 among small and medium-sized firms shows that 25 companies out of 72 have used mobile technology to improve their business processes [Pous+04]. All companies had mobile workers and almost 90 percent of them possessed a mobile phone, whereas only 30 percent had a PDA. Surprisingly, the most common way of communication between mobile employees and office workers was voice. Data transfer over a wireless network was not used, even in

these business segments where it could be expected, e.g. in construction companies. It was also noticed that double entry of data was not eliminated because mobile solutions were not introduced, although the survey participants were aware of the benefits of such solutions.

Another study was conducted between December 2004 and January 2005 among 95 enterprises in Germany. It showed that 55.2 percent of companies were using mobile technologies to access their information systems and 13.8 percent wanted to implement such access within the next two years [Faup06]. 64.6 percent of the enterprises with mobile data access planned to extend their systems in the future and 12.5 percent wanted to constrain their mobile activities. Figure 3.5 shows the detailed results of the study.

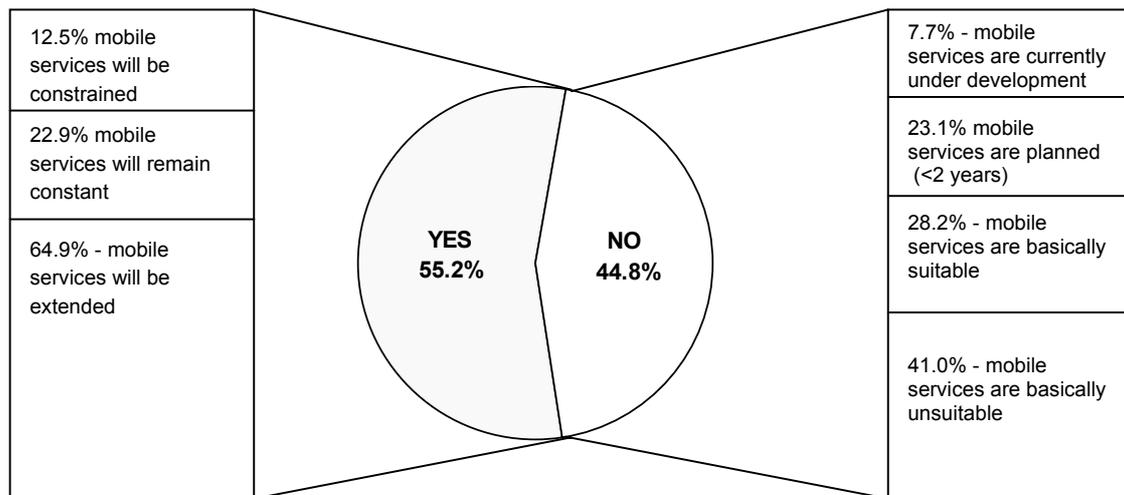


Figure 3.5: Deployment and usage of mobile technologies for mobile data access [Faup06, p. 61]

3.3 Adoption and acceptance of mobile Internet

The acceptance of mobile Internet and the availability of advanced mobile services differ widely across the world. In Japan, the introduction of i-mode in February 1999 has created a paradigm shift in mobile telecommunications and guaranteed mobile operators and content providers a great source of revenues. In Europe, mobile services based on the Wireless Application Protocol did not evolved as expected - mobile Internet and mobile business still lack prevailing users' acceptance. However, the failure of WAP or, more generally, mobile Internet in Europe and its amazing success in Japan cannot be attributed solely to technical differences between protocols applied to these services.

In Japan, the formerly government-owned telecom operator NTT DoCoMo decided to launch i-mode because the existing PDC network was running short on capacity. The company was confronted with a difficult choice. It could invest into a new and costly infrastructure in order to remain competitive or, alternatively, postpone the investments and start a program which would entice subscribers to use more data than voice, therefore exploiting the available network [cf. Tsuc00]. In Europe, the main players on the mobile market (Ericsson, Unwired Planet, Nokia) believed that they are unable to compete on the cost basis after the deregulation of the telecommunication market in European Union in 1992. Mobile Internet was seen as a great possibility for the creation of value-added services in

mobile networks⁴⁴. In 1997 Ericsson, Motorola, Unwired Planet, and Nokia started an initiative for the development of an open standard for mobile services and in April 1998 the first Wireless Application Protocol (WAP) specification was released.

I-mode achieved its success due to a very knowledgeable strategy of DoCoMo [cf. FuBa01; Funk00 for more details]. The company chose a revenue sharing model in which 91 percent of the fee for using a mobile service goes to a content provider, and only 9 percent to DoCoMo. Although the operator could deliver services only at the speed of 9.6 Kbps, it introduced packet switching and charged the customers for the delivered content. cHTML, which was chosen as the markup language for mobile services, facilitated the development of mobile content. Mobile services were furthermore divided into two categories: official and unofficial sites. The first type was integrated into the official mobile portal of DoCoMo that was used as the default start page in all phones. Moreover, in order to guarantee the high quality of provided services, DoCoMo used a special screening process and determined the content which could appear in the official menu. Japanese carriers had impact on manufacturers and were able to dictate technical requirements for produced handsets. Therefore, immediately after the start of mobile Internet, users could buy phones that supported new services. The most popular DoCoMo services were e-mail, ring tones, screensavers, and browsing. It is also worth mentioning that in 1999 only 13 percent of the Japanese population used the Internet [Cast+04, pp. 103-105]. This fact certainly contributed to the success of i-mode.

WAP, often deciphered by its dissatisfied users as “**Wait And Pay**” or “**Where Are the Phones**”, is perceived by many analysts as a very disappointing experience [cf. Cout+03; Endo03; Sigu01]. A report released by the International Telecommunications Union (ITU) in 2002 explains some of the reasons for the WAP failure. It mentions, as the most important factors, extended waiting periods for content downloading, ineffective billing models based on downloading time, the lack of content availability in WML and inappropriate (monochrome and very small) interfaces for viewing Web content [ITU02, p. 17]. Other authors emphasize that the number of handsets supporting WAP was very small by the time mobile services were introduced, because operators were not able to control handset makers [cf. Cout+03, p. 5; Sigu01, p. 32].

In the fall of 2000, Nielsen Norman Group conducted a field study on the acceptance of WAP services [NiRa00]. Chosen users got WAP phones and were asked to use WAP services and record their observations in a diary. The results of this study were very disillusioning, the report stated:

“WAP usability fails miserably; accomplishing even the simplest of tasks takes much too long to provide any user satisfaction. It simply should not take two minutes to find out the current weather forecast or what will be showing on BBC 1 at 8 p.m. [...] Considering that WAP users pay for airtime by

⁴⁴ Before starting a joined effort on a new, global standard for mobile Internet, each of the top players was working on its own standard. The most successful one was Nokia’s Smart Messaging protocol supported by a developed micro-browser, already introduced in March 1996 in the Nokia 8110 phone. For its services, Nokia was using a markup language called Tagged Text Markup Language (TTML). Similarly, Ericsson developed the Intelligent Terminal Transfer Protocol (ITTP) and Unwired Planet (currently Openwave) designed the Handheld Device Transport Protocol (HDTP) and Handheld Markup Language (HDML) [cf. Sigu01, pp. 4-7].

the minute, one of our users calculated that it would have been cheaper for her to buy a newspaper and throw away everything but the TV listings than to look up that evening's programs on her WAP phone." [NiRa00, p. 3]

70 percent of the users participating in the study stated that they are not going to use a WAP-enabled phone within the next year and 20 percent of them were convinced that they will not use WAP services within the next three years.

Experts in mobile technologies and mobile solutions are of the opinion that the failure of WAP services should not be attributed to immature or insufficiently developed technology [Sigu01, p. 15, Tee03, p. 65]. They usually blame unsuitable marketing and the lack of appropriate business models for the calamity⁴⁵. The WAP Forum ignored the opportunity to communicate the WAP concept to operators, content providers, and end-users. Instead of this, high expectations were created and propagated by the media. The vision could not be fulfilled because the operators from the WAP Forum failed to develop realistic business models that would encourage users and content providers [Sigu01, p. 15, Tee03, pp. 85-89]. Jenson [Jens03] suggests that WAP services were not successful because of the "WAP attitude" - it was assumed that mobile Internet will be the killer application regardless of the provided content or business models. These expectations were based on the previous success of the Web and the observed tendencies on the market for mobile devices. European mobile industry failed to understand customers, their needs, purposes, and reactions and tried to bend the users to the technology. Funk attributes the initial failure of WAP services in the USA and Europe to the lack of micro-payment systems that are a prerequisite for entertainment content like screen savers, horoscopes, and ringing tones [Funk04, p. 6]. European and American service providers focused rather on business content, like financial or shopping services that do not require micro-payment services.

After the initial failure, mobile services slowly recovered (cf. figure 3.6). Wireless Internet is particularly popular in Japan and Korea, in the U.S. and Europe the uptake has not taken place yet [Sriv04, p. 18]. Opinions and expectations about mobile business and mobile applications vary, depending on the company and time period. For example, in 2001 Jeff Bezos (the founder of Amazon.com) was convinced that in 5-10 years all electronic commerce activities will be conducted with the help of wireless devices [Clar01]. In 2005, Amazon scaled back its mobile activity because it has not brought as much benefits as expected. Wells Fargo that is offering banking services, abandoned mobile banking. While its Internet banking was used by 3.2 million people, only 2500 persons wanted to benefit from wireless banking [MaKhRa05]. On the other hand, eBay launched in 2004 in the UK a mobile service called eBay Anywhere⁴⁶. This service enables users to obtain SMS notifications about auction course and to place bids from a mobile phone. Google placed its search icon on the new Palm Treo 700w phone. A similar icon is also going to appear on several mass-market handsets that

⁴⁵ At the same time in Japan, KDDI launched a succesful WAP-service named EZweb. KDDI was also the first operator in the world using WAP 2.0 implemented over the CDMA network.

⁴⁶ Cf. <http://ebay.volantis.net/anywhere/>.

Motorola is planning to distribute in the second quarter of 2006⁴⁷. Business users consider mobile applications from the economic point of view: they look for quantifiable optimization potentials of business processes and try to maximize the return-on-investment of their projects [RaScFi05].

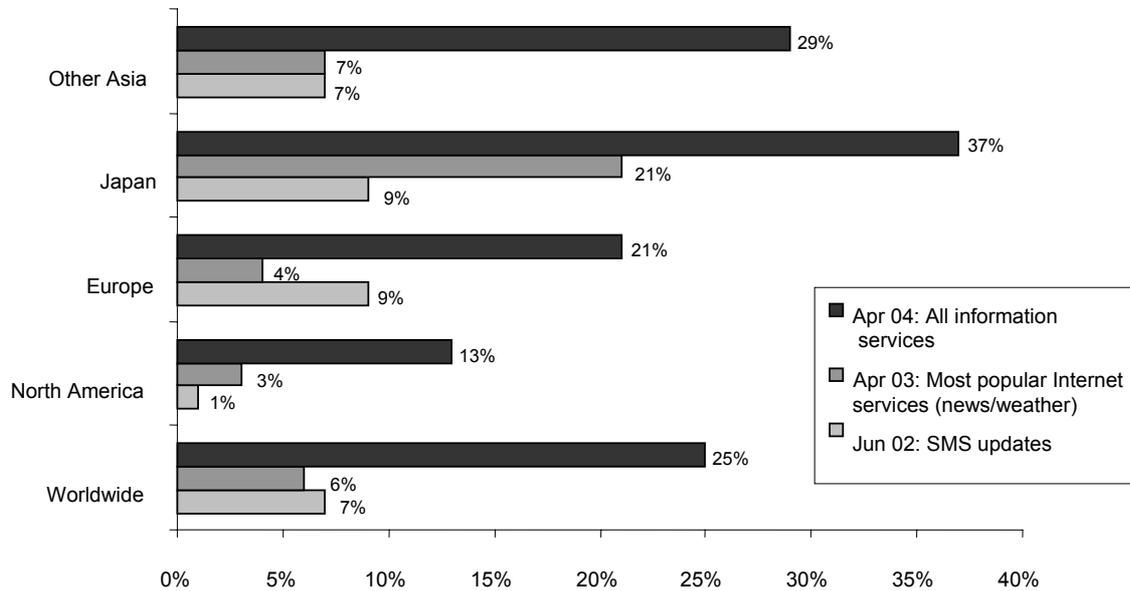


Figure 3.6: Usage of mobile information services (in % of multimedia phone users) [ATKe+04, p. 16]

The reasons for not deploying mobile solutions in enterprises are certainly different from these given by private customers, since the implementation of such solutions can have positive or negative impacts on the profits. Most enterprises are concerned with security and cost aspects as well as the insufficient maturity of the technology. According to a survey of 95 German enterprises, business users are mainly concerned about the following issues [Faup06, p. 62]:

- open technical security issues - 74.4 percent of asked enterprises
- insufficient quality of mobile services - 50.6 percent
- costs are higher than the potential benefits - 46.3 percent
- insufficient standardization of mobile technology - 40.3 percent
- expensive integration into the existing IT infrastructure - 39.2 percent
- expensive integration into the existing operations/organization - 37.5 percent
- unclear juridical aspects (for example, liability in case of data losses) - 34.9 percent
- too little employees with sufficient qualifications/knowledge - 27.9 percent
- open security questions of non-technical nature - 25.9 percent
- concerns with regard to health (for example, high radiation) - 12.2 percent.

Although various companies may represent different views on future prospects of their mobile solutions, researchers willingly investigate factors that lead to the rejection or adoption of mobile applications. The significance of old arguments explaining the WAP failure fizzled due to the

⁴⁷ Cf. <http://www.wirelessweek.com/article/CA6297580.html>.

improvements in network technologies and the emergence of new pricing models. However, the adoption of mobile services is still relatively low [RaScFi05]. Different surveys deliver disparate reasons for this fact. They try to apply different models and theories to explain this phenomenon. The Diffusion of Innovations Theory [Roge95], the Technology Acceptance Model [Davi89], the Theory of Planned Behavior [Ajze91], the Theory of Reasoned Action [AjFi80], the Unified Theory of Acceptance and Use of Technology [Venk+03], and many others are used to explore the acceptance and use of mobile services. Although the categories of mobile users distinguished in different publications may vary with the applied theory, the reasons for negative attitudes towards mobile services are similar across different studies.

The research on mobile adoption emphasizes that the traditional Internet is the most dangerous rival of mobile browsing. Users consider the Internet and mobile Internet as substitutes – they are ready to go online using a mobile phone only if they are unable to access the same content using a PC (e.g. while traveling) [CoBIDa05; Dams+05]. Otherwise, they are not willing to pay for browsing with the help of a mobile device. They perceive the traditional Internet as a more convenient and cheaper solution.

Users, who were asked to evaluate mobile solutions in various surveys, complain about their insufficient usability related to sensory and functional needs. They mention the following factors that discourage them from using mobile services or are obstacles in the use of wireless applications [Knut05; MaKhRa05; Pede05; RoPi04]:

- poor mobile content
- inconvenient devices (difficult navigation and browsing)
- insufficient download/upload quality and data transfer
- poor stability, speed, coverage of the network
- security aspects
- high grade of difficulty/complexity to utilize the services, problems with configuration and set-up
- too much advertising, and finally
- high costs.

The surveys were performed in Europe and in the U.S. Some of them have national character and apply only to one country (e.g. Finland). Similar results, however, were delivered by international studies [AtKe+04; AtKe+05]. Figures 3.7 and 3.8 present the results of the so-called Mobinet Index. This is a study which examines trends in the mobile data and Internet usage, performed by A.T. Kearney and Cambridge University's Judge Institute of Management. It is based on interviews with around 5000 mobile phone users from 14-21 countries⁴⁸ and is therefore free from national biases.

The study indicates that prices of mobile services are above consumers expectations. One-third of mobile phone users were worried about the cost of mobile data, and about half of them were not willing to pay more than \$5 a month for it (result adjusted for the U.S. purchasing power parity). One of the

⁴⁸ In 2004 users from Australia, Brazil, Canada, China, France, Germany, Italy, Japan, Mexico, South Korea, Spain, Sweden, the United States, and the United Kingdom participated in the study. In 2005 Portugal, Denmark, Finland, Czech Republic, Poland, Russia, and New Zealand were added to the survey.

main concerns was the quality of the content offered on the Internet. More than one-third of mobile users named the availability and quality of mobile content as the main impediment for using mobile Internet - a considerable increase over 2004 (cf. figure 3.8). 27 percent of users, who already browsed mobile Internet, were not satisfied with the offered quality. Therefore, they were not keen on using wireless Internet more often (cf. figure 3.7).

Although the proportions differ between mobile Internet users and non-users, they all named similar reasons for the bad perception of mobile solutions. Both tech-savvy and less-experienced users were concerned about the technical complexity of offered applications and therefore demanded simpler, out-of-the-box operating experience. High costs are still an important obstacle in mobile browsing - questionnaires showed that subscribers with mobile accounts subsidized by their employers were more likely to use mobile Internet than users who had to pay for mobile access individually [Metr05]. The Mobinet study also revealed that the marketing of new data services is not done appropriately. Most users focus on purchasing basic voice services at the lowest price possible and do not even consider the use of mobile data services.

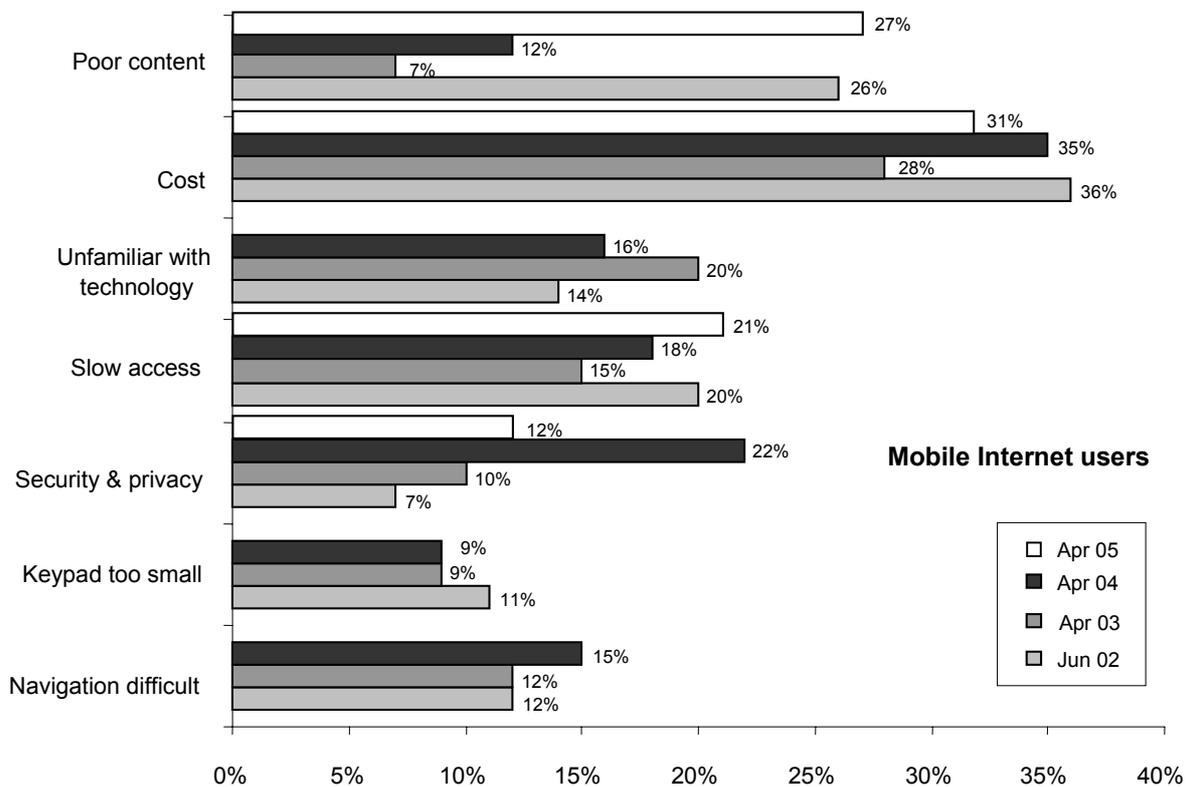


Figure 3.7: Potential problems with the use of data services (as identified by mobile Internet users) [ATKe+03; ATKe+04]

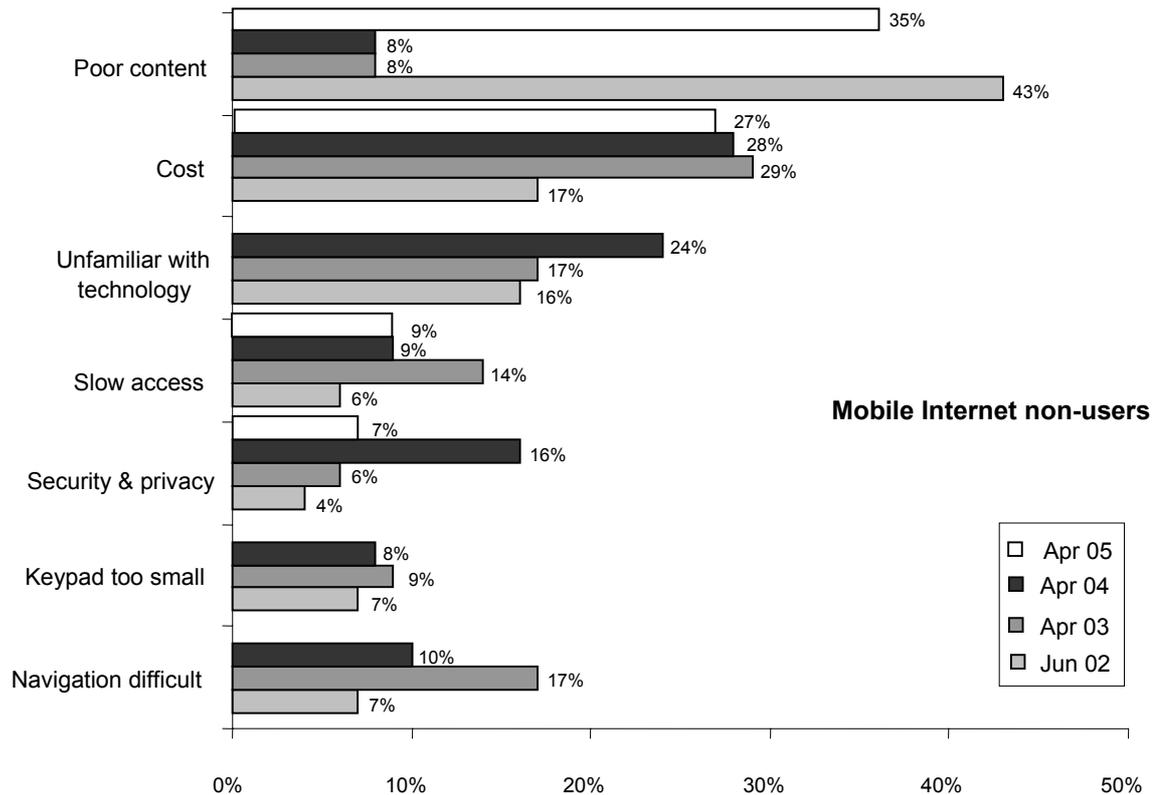


Figure 3.8: Reasons for not using mobile Internet [ATKe+03; ATKe+04]⁴⁹

Findings of the research on mobile services adoption suggest that the services need to deliver context-specific values to private and business users. Anckar and D’Incau identify five settings in which mobile value can be created [AnDi02]. Under mobile value they understand the benefits arising from the mobility of devices in connection with immediate access to information. These settings are as follows:

- a) time-critical arrangements - mobile solutions can be applied in situations which arise from external events and require immediate actions (alerts for stock traders or for managers controlling flows in supply chains).
- b) spontaneous decisions and needs - do not result from external events and cannot be foreseen. They are related to products and services that do not have to be carefully considered in order to undertake a purchasing decision. Spontaneous needs can be related to entertainment, efficiency gains or can be time critical.
- c) entertainment needs - can be observed in situations where the user wants to have fun and does not have access to traditional wired entertainment appliances such as TV sets.

⁴⁹ In 2005 not all reasons were evaluated.

- d) efficiency ambitions - resulting from the possibility to use “dead spots” during a day to work and to increase the overall productivity. Such ambitions can be particularly related to mobile workforce, for example, sales force employees.
- e) mobile situations - situations in which the user does not have any access to another information medium and has to accomplish a particularly important task or in which a mobile device can be of additional use while on the move (e.g. roadside services).

Mobile solutions should be better designed and support current day-to-day practices. To increase their revenues and market share, operators should focus on marketing models that would encourage the use of services, for example, by applying new pricing models along with improved content and customer interfaces. The key marketing challenge is to create targeted value propositions addressed to different segments of consumers (e.g. segmentation based on voice and data usage). To attract new customers, operators should perform market testing, implement usable content-rich services, and offer low-price packages. Unless these issues are taken into consideration, the users will judge mobile services by their first impressions and subjective feelings/opinions of other consumers. They will never take a chance to use mobile services themselves.

Some analysts emphasize that the major reason for the failure of mobile Internet outside of Japan and Korea is the lack of solution for the so-called “startup problem” [Funk04a]. The success of products that exhibit network effects depends on the number of users. If firms do not solve the startup problem by creating a critical mass of users, innovation stops and products may disappear from the market. This problem is critical for mobile Internet because service providers, content and application providers, handset manufacturers, and other technology providers have to coordinate their activities to attract users.

In Europe service providers do not want to share their revenues with content providers. Therefore they do not offer users easy and cheap access to push-based Internet mail and content via the input of URLs. In this way, service providers want to protect their SMS revenues and revenues from entertainment contents. In 2004 Credit Suisse argued in its report: “We believe that operators must still address issues of walled gardens or open content and also how they charge for WAP access. [...] i-mode operators have adopted a very open approach to content, giving users easy access to any sites. By contrast, Vodafone, Orange, and T-Mobile have a much closer control over content, allowing them to control quality but also restricting the opportunities for content providers” [Funk04a]. Despite improvements in transfer rates speeds, network operators still restrict the content offered, for example by encouraging mobile users to view only certain services and signing exclusive contracts with service providers (e.g. the “web'n'walk“ service of Google and T-Mobile for German and Austrian owners of handsets) [FAZ05].

The examples of Japan and Korea show that the development of mobile content and push-based mail services are positively correlated – offering mail services can encourage users to browse in the mobile Internet. Additionally, large service providers like Vodafone and large manufacturers like Nokia struggle for control over phone specifications. Government regulations seem to be the only way to force service providers to adopt a more open approach towards mobile Internet.

4

Requirements and guidelines for device-independent content presentation

Adaptation approaches have to fulfill various requirements to gain widespread acceptance of end users and developers. They should assure good software quality as defined by the commonly known ISO quality standard and have to possess characteristics that would guarantee rich user experience for customers with different access mechanisms. Furthermore, content authors should not be overwhelmed with advanced language features that they are not able to understand and apply. Aspects like costs of approaches, efficiency of use, or deployment possibilities should not be neglected when specifying the requirements for adaptation frameworks.

This chapter focuses on many different factors that have an impact on the quality of device-independent solutions. Section 4.1 defines the term “adaptation” and introduces its essential characteristics. It focuses on three basic aspects of adaptation: its kind, subject, and quality. This section also demonstrates some basic techniques for adapting application’s elements (layout, structure, navigation, etc.). The next section introduces the so-called Device Independence Principles (DIPs) identified by the W3C Consortium with regard to the user’s, author’s and delivery perspective. Section 4.3 outlines general software quality factors according to the ISO 9126 norm. It provides an overview of basic considerations and challenges for people participating in the process of device-independent content authoring and management. The presented norms and principles have diverse levels of abstraction and take into account various aspects of device-independent approaches. They can therefore be regarded as a good basis for building a comprehensive framework for the evaluation of existing and new authoring solutions. Last but not least, section 4.4 lists the most important guidelines that may be very useful in the process of designing and developing mobile applications.

4.1 Essential functionality of adaptation and adaptation methods

Research on the adaptation of Web pages started with the emergence of the adaptive hypermedia concept in the 1990s [cf. BoHoSc90; BrPeZy93]. Adaptive hypermedia systems were able to adjust themselves to the needs of users. They possessed a model of user goals, preferences and knowledge. The adaptation was further enhanced by applying data about usage (e.g. usage frequency of particular sites or action sequences in browsing) and environment (software and hardware components, user’s location). Brusilovsky and Kobsa [Brus01; KoKoPo01] distinguish in their research three basic adaptation types: adaptation of content, adaptation of presentation and modality (from image to text, from text to audio, etc.) and adaptation of structure.

With the emergence of heterogeneous devices, the necessity of adaptation became inevitable. Its main goal was not to make Web-based systems more user-oriented, as in the adaptive hypermedia concept, but to enable the presentation of content on different devices. The word “adaptation” originates from Latin “ad aptare” (to fit to). It stands for a change in partial or entire characteristics of an entity under consideration and is always based on some context [Hol92]. In the context of device-independence, the term adaptation describes “a process of selection, generation, or modification that produces one or more perceivable units in response to a requested uniform resource identifier in a given delivery context” [W3C03a].

Adaptation can be characterized by its kind, i.e., changes that should be performed, the subject of adaptation, and the quality of adaptation [cf. Kapp+02, pp. 2-3]. With regard to the kind of adaptation, one should consider its effect, complexity, and universality. Considering the effect, adaptation may enhance an application by adding new parts to it. It may also reduce an application by removing its fragments or perform some transformations (e.g. replacing multimedia content with descriptions). Adaptation can be simple and atomic or complex, consisting of a set of adaptations. With regard to universality, adaptation may be application-specific and useful only for certain types of applications or generic and universally applicable (e.g. changing the resolution of images).

The quality of adaptation can be determined by its dynamism, automation, visibility, and reusability. Adaptation may be static or dynamic. In static adaptation, the adapted content is defined at design time (e.g., two versions of the same picture for different bandwidths are prepared). In dynamic adaptation, the content is generated at application’s runtime. Depending on the user’s influence on the adaptation process, one can distinguish between manual, semi-automatic, and automatic adaptation. Manual adaptation means that the user can determine the kind of adaptation. In semi-automatic adaptation, the user can influence the adaptation process, for example, by specifying her/his preferences and choosing appropriate versions of content. Automatic adaptation is performed without the user’s interferences. Moreover, the adapted content can be visible globally or only by certain groups of users. Last but not least, adaptation may be performed from scratch, basing on the original version of content, or incrementally. In the last case, it is applied to the content that was already transformed in preceding adaptation steps.

Subjects of adaptation are style, layout, content, structure, navigation, application interactions, and transmission of data [cf. W3C04]. Style refers to the visual appearance of text (i.e. colors, fonts) and layout features of textual elements. Layout of content is the visual or temporal arrangement of particular components. Content is the information relevant to the users (in the form of text or graphics). Structure describes relationships between parts of delivered content and is influenced by navigation that offers the user a possibility to move between presentation units. Application interaction is the manner in which the user transmits information to a server with the help of the characteristics of delivered content in order to influence subsequent content delivery (e.g. form processing). Transmission of data refers to the method of transferring information (e.g., transportation of data processed on the client).

Adapting the style and layout of documents

In the adaptation of style and layout, the content of objects (e.g. tables, paragraphs, etc.) remains the same, while the format and layout of the objects change. A special type of style adaptation is the varying input/output modality in multimodal systems. In such systems, one can use alternative access methods and interact with an application in a variety of ways, using speech, keyboard, or stylus [cf. AnBaSh01]. For example, the user may receive information in the form of a graphical presentation enhanced with sound and respond to it with her/his voice and a touch screen.

The presentation style depends on the features of devices and on user preferences. Users can specify, for example, what colors and font types they prefer, or which arrangement of elements they find optimal. Device types also influence the presentation of content because of their often limited capabilities to render colors or scarcity of available font types and sizes. Style adaptation is usually achieved with device-dependent style sheets (e.g. in the form of CSS or XSLT)⁵⁰.

In layout adaptation, spatial and temporal layouts are considered. Spatial layout specifies the arrangements of areas in which content appears (e.g. vertically, horizontally). Temporal layout refers to voice or media applications and controls the presentation of elements in time. Furthermore, parts of the content may be made invisible. The layout can be defined, for example, by using SMIL that provides layout notions based on the `<layout>`, `<root-layout>` and `<region>` elements.

```
1. <smil xmlns="http://www.w3.org/2001/SMIL20/">
2. <head>
3. <layout>
4. <root-layout width="320" height="480" />
5. <region id="a" top="5" bottom="100" />
6. <region id="b" top="200" bottom="280" />
7. </layout>
8. </head>
9. <body>
10. <text region="a" src="text.html"/>
11. <text region="b" src="additional_text.html"/>
12. </body>
13. </smil>
```

Listing 4.1: Layout specification in SMIL [W3C04]

Listing 4.1 demonstrates a simple layout definition in SMIL. Within the `root-layout` (320 pixels in width, 480 pixels in height), two regions with full width of the `root-layout` are defined. In the `body` element, two `text` elements assign the content to the regions, providing a mapping between the content and the layout.

⁵⁰ For more details cf. sections 6.1 and 6.3.

Adapting the structure and navigation of documents

In the adaptation process, the transformation of structure and navigation elements is particularly important. Web documents can contain internal and external elements that are linked to other documents (e. g. media objects). External elements described correctly (for example, by annotation tags) can be split into atomic units and inserted into new structures. The structure of elements without description can only be adapted using heuristics⁵¹.

Structure adaptation

Aggregation and decomposition play a central role in the process of structure adaptation. In aggregation, content fragments from one or more sources are collected to form a single fragment or page. Individual content fragments contain certain structures and the aggregation process leads to a new structure.

In decomposition, authored units are divided to create a new set of presentation units, suitable for a particular delivery context. A decomposition approach has to specify how the units will be divided and referenced from each other (navigation mechanism). Pagination is the commonly known method for decomposition of content that possesses linear structure. Different pagination techniques exist. Page breaks, window/orphan control, "keep together/keep with next", sectioning, and regions are frequently applied in pagination solutions [W3C04].

Page Breaks

In the page breaks technique, the author marks the point where the content should be split. The position of the mark should depend on contextual information such as the display size of a device.

Widow/Orphan Control

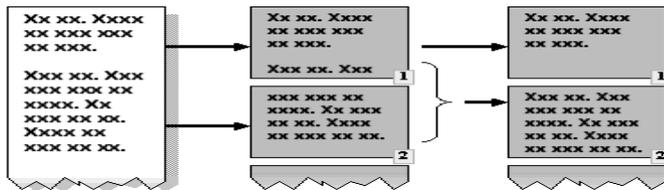


Figure 4.1: Widow/orphan control pagination [W3C04]

In the technique called widow/orphan, control paragraphs are divided in a way that tends to avoid leaving a single line on a page, separately from the rest of the content. A "widow" is defined as one line from N lines on one page while the rest (N-1 lines) remains on the second page. Similarly, an "orphan" occurs when N-1 lines are placed on the first page and the last line slips to the next page. Figure 4.1

⁵¹ Heuristics describe a set of rules developed to solve a problem when appropriate algorithms cannot be designed.

illustrates the concept of the widow/orphan control - the last line from the first page (the widow) is simply moved to the next page.

"Keep Together" and "Keep With Next"

In the "keep together" pagination technique, paragraphs are not divided and are not separated from the subsequent paragraphs. "Keep with next" is usually applied to headings that are not separated from the content describing the heading, but rather moved to the next page, if necessary.

Sectioning

In the sectioning approach, the author applies headings to identify sections within the content. As figure 4.2 presents, the headings are used as division points in the decomposition process. The `<h1>` tags in HTML can be used to mark sections. In XHTML 2.0, the `<section>` element identifying document sub-trees was introduced.

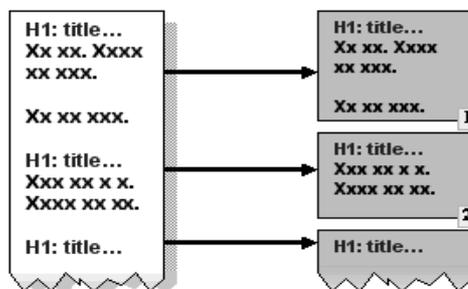


Figure 4.2: Sectioning pagination technique [W3C04]

Regions

Regions are fragments of content identified by a starting and an ending mark. In XHTML and HTML, the `` and `<div>` tags can be applied as marks. In regions, the "id" attribute can uniquely specify fragments in a document. Regions may be furthermore nested. The concept of pagination according to regions is displayed in figure 4.3.



Figure 4.3: Regions pagination technique [W3C04]

Adaptation of navigation structure

The adaptation of navigation structure by modifying an existing structure or by creating a new one from the content is a difficult task. It can be performed automatically only in some situations if the navigation paths are determined (e.g. creating sitemap from all links, generating table of contents from headings). Typical navigation structures are tables (lists), menus (hierarchies), and links (previous/next). A navigation structure may be generated by using existing links or by creating links from significant parts of the content (e.g. titles, headings, etc.). Classic navigation structures are tables of contents, navigation menus, and next and back links [cf. W3C04 for more information].

Table of contents

A table of contents is a list consisting of headings, structured according to the order of appearance, the relative importance, or the alphabetical order of headings. The headings are usually displayed as links, offering the possibility to navigate to the referenced portions of the content.

Navigation menu

A navigation menu is a list that contains links to the content of internal or external pages and helps to find relevant documents. This menu can be static or dynamic (e.g. drop-down form elements, collapsible navigation trees) and its complexity depends on device features. A navigation menu can represent the whole site, or show only some contextually relevant documents on the site and provide an additional link to the home page.

Next and previous/back links

The concept of a menu in the form of next and previous/back links refers to the navigation path, created while a user navigates through a site. The next and previous links are generated relatively to the navigation path. Clicking on the previous link redirects the user to the page that was viewed prior to the current page. Clicking on next link allows the user to return to the established path. In the next and back navigation, the next page is determined by the author and the back page refers to the last page in the navigation path.

Adapting the content of documents

The content of Web documents can be decomposed into three categories: text, images, and multimedia elements. Parts of documents can be adapted by changing the properties of objects (for example, the resolution or color depth of an image) and data representations (file formats), by replacing some objects with other ones, or by eliminating content fragments.

Text usually possesses some structure (title, headings, sections, abstracts, meta information, etc.) describing its semantics. Text can be adapted by extracting only the most relevant fragments from it, or by splitting the information into many fragments, connected by an appropriate navigation structure.

In adaptive hypermedia systems, from which device-independent approaches benefit, adaptation of content can be achieved with page or fragment variants, adaptive stretchtext, fragment coloring, and

adaptive, natural-language generation [cf. Brus01; KoKoPo01 for more information]. The page-variant approach is simple but inflexible and costly: for each adaptation, considered as necessary, a separate page has to be created. In the fragment-variant approach, separate variants are authored for each adaptive page fragment. The fragments can then be combined dynamically, forming one page. A frequently applied technique consists in insertion or exclusion of optional fragments, depending on user preferences or device features. A simple technique to implement the optional text approach is to associate a condition with each optional chunk of text that determines whether inclusion or exclusion should be applied to this fragment. In device-independent approaches, the inclusion/exclusion of content is often achieved with SMIL [W3C05b].

In the fragment-coloring method, certain elements of the presentation are marked out as being important or interesting to the users. Authors define as “stretchtext” the text that can be extended or collapsed by clicking on it with the mouse. Depending on user preferences, stretchtext can be automatically extended for certain users. In adaptive, natural-language generation, alternative text descriptions are created for different users. For example, text templates with slots can be filled with descriptions with different complexity grades, based on the user’s level of expertise.

Image adaptation

In image adaptation, the size, resolution, color depth, and file format often have to be modified. The main goal of image adaptation to capabilities of mobile devices is to decrease the file size while preserving the quality of a picture. Thumbnails with links to the original image or slightly reduced images can be created. If the available bandwidth is too low or the display size is too small, images can be replaced by textual descriptions. In image adaptation, three adaptation methods can be distinguished: static, progressive, and dynamic adaptation [LeDe05; LeGe01].

Static adaptation involves the creation of multiple versions of images that would correspond to different characteristics and requirements of devices. The version that matches the capabilities of a device is then sent to it. This method eliminates the problems with server latency because images do not have to be processed at runtime. On the other hand, it results in significant storage overhead and high maintenance cost.

In dynamic image adaptation, images are transformed according to device characteristics for each service request. In this approach, mobile applications are more flexible and can offer customized image versions in terms of content quality, formats, etc. Due to the fact, that a special image version has to be prepared for each response, dynamic adaptation is slower than static adaptation and can lead to longer answer times.

In progressive adaptation, a special version of an image is generated and sent to all users. The user can download the whole version at once or can receive it in parts. The quality of the image improves as the user obtains more portions of the data transmitted. This method can lead to over-utilization of networks if redundant data is transmitted. In this approach, the quality of images may also be worse, because a single image version has to satisfy the requirements of all the devices.

Adapting the transfer of documents

Despite recent improvements in this domain, mobile connections still rely on low bandwidth, high latency, error-prone, and expensive networks. Instead of sending the whole content to the device and adapting it on the client, documents may be adapted before the transfer. In case of network errors, the proxy can prevent possible data losses by caching the data for further use. Furthermore, transfer adaptation may take into consideration the maximal size of a WML deck or the allowed size of downloaded Java ME applications.

4.2 Device Independence Principles

To enhance user experience related to the content delivered to various devices, the W3C Consortium specified certain principles for device independence [W3C03a]. These principles should not be regarded as a strict set of requirements. They should rather serve as guidelines for the design of applications based on existing markup languages, during the development of adaptation tools, in the process of extending existing markup languages, or for the design of new markups. According to these principles, content adaptation aspects can be viewed from three different perspectives: user's perspective, author's perspective, and delivery perspective.

User-related principles

Generally speaking, the user would like to interact with the Web using various devices and via many access mechanisms. From the user's perspective, two most important aspects of content adaptation are: device-independent access to the same functional presentation of content (DIP-1) and device-independent Web page identifiers (DIP-2)⁵². Device independent access means that the user is able to obtain the equivalent application functionality regardless of the device type. Obviously, the presentation will not be the same on every device and its quality may vary. The intended functionality of a page should be associated with only one Web page identifier, and should apply to all presentations obtained from the same Web page identifier, no matter what the access mechanism is. For example, a user who enters one URL for a weather service in a browser on his/her PDA and in a browser on his/her WAP-enabled mobile phone should see the forecast for a chosen city in the form of texts and/or images. On a screen of the mobile phone, the temperatures and WBMP images symbolizing the weather conditions will be displayed. On the PDA, the user will get colored JPEG pictures and more detailed information about the weather, due to the larger screen. Navigation structures and pagination will be different, but the user will be able to see the weather forecast for the same cities. User experience in the form of messages like "cannot display images" or "deck size is too large" are considered as a delivery of a non-equivalent functionality.

The W3C recommends the use of device-independent Web page identifiers and tries to keep the Web unfragmented. The consortium objected to the specification of a new top-level domain (TLD) ".mobi"

⁵² The abbreviation DIP stands for device independence principles.

that would identify only mobile sites. Existing TLDs (.org, .edu, .com, etc.) refer to the source or organization type that is in charge of the content (company, educational organization, etc.) and not to the type of supported devices [Perl05]. The creation of a mobile top-level domain would therefore separate the content displayed on regular computers from the content displayable on mobile devices, and would result in two parallel Webs - the traditional one and a new, mobile Web.

Author's perspective

From the author's perspective, it should be possible to provide a functional presentation in response to a request associated with a specific Web page identifier, in any given delivery context that has an adequate access mechanism (DIP-3). This principle is simply a restatement of the aforementioned principles from the author's perspective. The adaptation process should provide a presentation that allows the user to successfully access a Web page and to get information from it or to complete the interaction intended. If a functional presentation of an application cannot be delivered due to inherent limitations in the access mechanism, an appropriate error message should be shown to the user (DIP-4). Additionally, it should be possible to provide a customized and harmonized presentation depending on device features (DIP-5). The author should be able to control the presentation on different devices. It is unrealistic to expect from the author that he/she will create different presentation data for each delivery context. Whenever possible, the authored source content should be reused across multiple delivery contexts.

Delivery perspective

Delivery-related adaptation principles encompass the characteristics of delivery context and delivery preferences. The term "delivery context" refers to a set of attributes that are given for a particular delivery environment. The adaptation software should be able to associate the characteristics of the delivery context with a request for a particular Web page identifier (DIP-6). Unless the characteristics of the delivery context were made available to the adaptation process, it is not possible to find out whether a specific presentation of content can be delivered in this context, or how to generate a suitable presentation. The user should be able to provide or update any presentation preferences as part of the delivery context (DIP-7). The delivery context should enable the creation of context-aware applications. If the user provides presentation preferences, they may be used in the adaptation process to offer a more suitable presentation, after taking into account the constraints of the network and the device in question. The process should allow the user to obtain the most appropriate presentation with regard to her/his abilities and all circumstances.

In device-independent approaches, capabilities of devices are the most important part of the delivery context. Relevant features include screen characteristics, supported languages and formats, input/output capabilities, speed of network connections, maximum size of the downloadable content, processor and memory capabilities. Screen capabilities refer to the screen size, supported font types and sizes, as well as to displayable colors. Mobile devices can support various markup languages such as WML, XHTML, HTML, CHTML, HDML, or Java ME and different multimedia types (audio and video formats, various types of images). Input capabilities of mobile devices are different from the QWERTY-keyboards, used with traditional computers. Most devices do not offer a full keyboard, but rely on

touch-sensitive screens or phone number keyboards with 12-15 buttons. Furthermore, mobile devices can provide only one output type (e.g. textual/graphical) or support different output modalities (e.g. voice, video, text, etc.). The connection speed, acceptable payload size, processor power and memory capabilities are features that can also influence the user experience, and should be considered in the process of application design. Furthermore, a document or application exceeding the maximal allowed size or free memory cannot be displayed on a device and advanced, resource-intensive Java ME applications should not be provided for devices with limited processor or memory capabilities.

4.3 Quality model and authoring challenges for device-independent content presentation approaches

The quality of a software product is defined in different ways by various authors [cf. CoLo99; Niel93; Pree+94; etc.]. The ISO/EIC 9126 standard [ISO01] is part of the most important norm for quality assurance - ISO 9000. According to this standard, quality is defined as “the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs” [ISO01]. ISO/IEC 9126 standard distinguishes between two aspects of quality: external/internal quality and quality in use [ISO01; ISO03; ISO03a; ISO04]. Internal quality concerns features of non-executable fragments of software during the development process and is measured in terms of quality of the deliverables, e. g. basing on the source code of a prototype. External quality focuses on the behavior of a system and requires the execution of the product in its intended environment. Quality in use deals mainly with user’s perception and acceptance of the software and includes four features [ISO04; ISO98]:

- effectiveness – the product should enable users to achieve specified goals with accuracy and completeness
- productivity – the ratio of used resources (expenditures) to achieved benefits
- safety – understood as possible risk of harm to people, business, property, or the environment
- satisfaction - the capability of software to satisfy user’s needs.

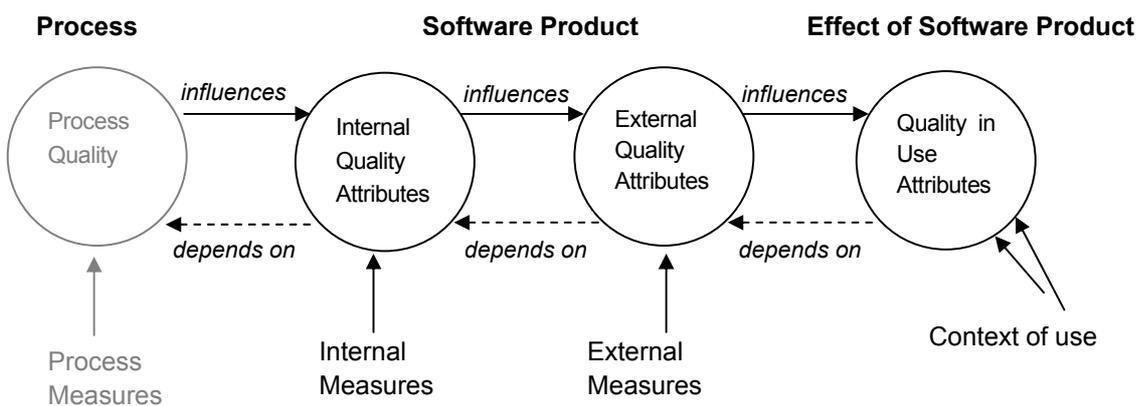


Figure 4.4: Quality in the lifecycle [ISO01]

The main objective of the ISO quality norm is to guarantee that a product has the required effect in a particular context of use. Examples of contexts of use are presented in table 4.1. This goal can only be achieved by evaluating the quality of software during its entire lifecycle as depicted in figure 4.4.

Possible metrics for internal/external quality as well as for the quality in use are introduced in Appendix 2.

Component	Relevant data
Users' characteristics	<ul style="list-style-type: none"> – Psychological attributes including cognitive style (e.g. analytic vs. intuitive, attitude towards the job, motivation to use the system, habits, etc.) – Knowledge and experience including native language, typing skills, education, system experience, task experience (knowledge of the domain), application experience (knowledge of similar applications)
Tasks characteristics	<ul style="list-style-type: none"> – Frequency – Duration and importance – Task flexibility/pacing – Physical and mental demands – Complexity as perceived by the user – Task structure (highly structured vs. unstructured) – Task flow including input/start conditions, output and dependencies – Relation to business workflow
Technical environment characteristics	<ul style="list-style-type: none"> – Hardware capabilities and constraints – Network connection – Operating system – Supporting software
Physical environment	<ul style="list-style-type: none"> – Noise level, privacy, ambient qualities, potential health hazards, safety issues
Organizational environment	<ul style="list-style-type: none"> – Structure of the operational teams and the individual staff members' level of autonomy – Work and safety policies – Organizational culture – Feedback to employees with regard to job quality
Social environment	<ul style="list-style-type: none"> – Multi- or single-user environment – Degree of assistance available – Interruptions (e.g. disturbing environment)

Table 4.1: Exemplary context of use attributes [ISO98]

The internal and external quality attributes for software products are further divided into a set of sub-factors, as displayed in figure 4.5. The ISO 9126 norm provides requirements that helps to evaluate different products.

The quality factors can be explicit or implicit in nature. Explicit quality attributes refer to the documented functional and performance requirements as well as to specified development standards. Implicit requirements are those that are not formally documented but expected by users. The basic quality factors are functionality, reliability, usability, efficiency, maintainability, and portability. They are briefly explained in the following section.

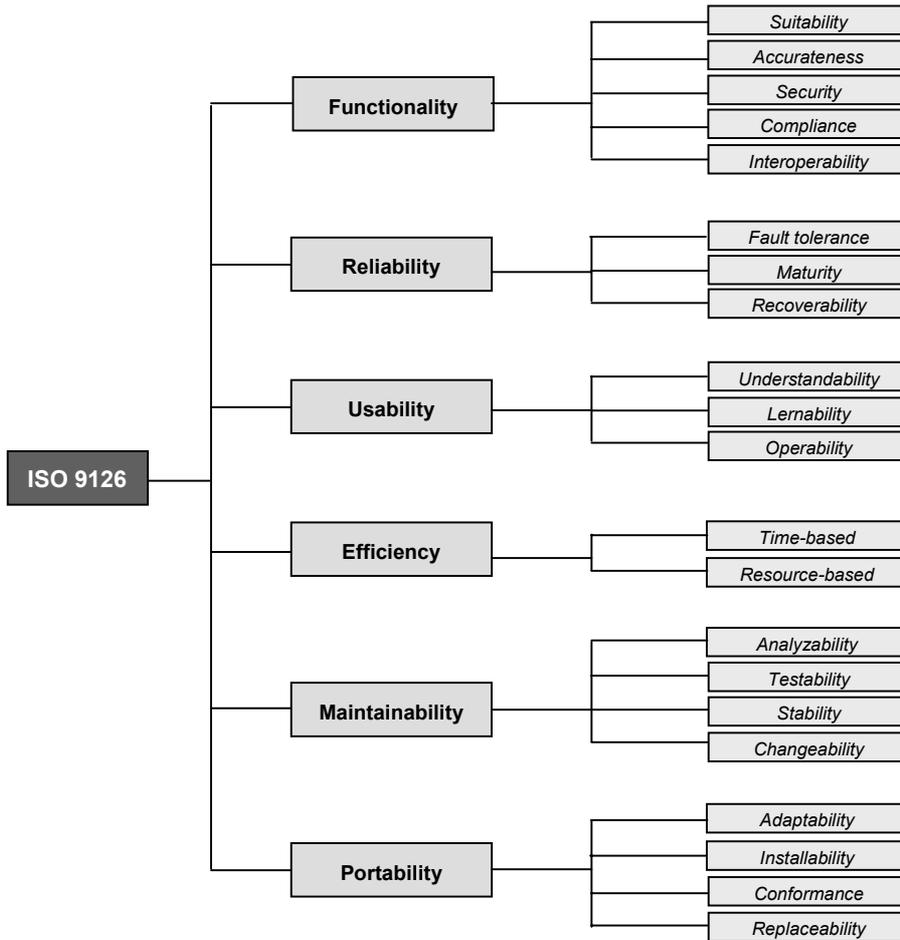


Figure 4.5: ISO 9126 quality standard [ISO01]

Functionality

Functionality is “a set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs” [ISO01]. Functionality can be divided into five sub-factors: Suitability means that the functionality of an application is appropriate to accomplish a specified task. Accurateness refers to the attributes of software that guarantee that the results or effects obtained by the user are correct. Interoperability signifies the ability of an application to interact with other systems. Compliance means the adherence to standards, conventions, or regulations. Security is defined as the ability to prevent unauthorized access to programs and data.

Reliability

Reliability is “a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time” [ISO01]. It consists of three sub-factors: maturity that refers to the frequency of software faults, fault tolerance, defined as the ability of software to deal with software faults, and recoverability that is understood as the capability to recover lost data in the case of a failure.

Usability

Usability is "a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users" [ISO01]. In usability, three sub-factors can be distinguished: understandability describes the user's effort necessary to understand the logical concept of an application and its applicability. Learnability refers to the effort required to learn the application and operability means the user's effort and time invested in operations and operation control. Usability can be measured in terms of physical and intellectual skills needed to learn a software and to work with it, or the time necessary to get to know the software.

Efficiency

Efficiency is "a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions" [ISO01]. Efficiency is divided into time-related and resource-related efficiency. Time-related efficiency refers to temporal aspects of the interaction with an application while resource-related efficiency takes into account the required resources.

Maintainability

Maintainability is "a set of attributes that bear on the effort needed to make specified modifications" [ISO01]. Maintainability consists of four criteria: analyzability, which is a measure of the effort required to diagnose deficiencies or failures and for the identification of necessary modifications. Changeability measures the effort needed for modification, fault removal, or environmental change. Stability is defined as the acceptance of unexpected modification effects. Testability refers to the effort connected with the validation of undertaken modifications.

Portability

Portability is "a set of attributes that bear on the ability of software to be transferred from one environment to another" [ISO01]. In portability, four additional factors can be distinguished: adaptability means the possibility to adapt the application to different environments/platforms without additional actions or means. Installability is a factor measuring the effort for installing the software in a new environment. Conformance relates to the conformance of the software to portability standards or conventions. Replaceability is the effort and the possibility of replacing an application with the evaluated one.

Quality in Use Integrated Measurement

In order to evaluate a software product, it is important to focus on software usability and quality in use factors. These two aspects can be determined with the help of an enhanced usability model called Quality in Use Integrated Measurement (QUIM) [Abra+03; DoSe01; SeDoKI05]. This model is based on ISO standards and previous research in the area of usability and quality in use. It distinguishes 10 usability factors and 26 sub-factors referred to as criteria [SeDoKI05, pp. 11-13].

The usability factors from the QUIM model are as follows [SeDoKI05, p. 11]:

- 1) Efficiency – defined as appropriate amounts of used resources in relation to the effectiveness
- 2) Effectiveness – a user's ability to achieve specified goals with accuracy and completeness
- 3) Productivity – the level of effectiveness in relation to various resources, for example time, user efforts, financial costs, etc.
- 4) Satisfaction – subjective level of satisfaction expressed by users
- 5) Learnability – ability to achieve specified goals easily, the product should be easy to understand and learn
- 6) Safety – the capability of the product to limit the risk of harm
- 7) Trustfulness – users should trust the system/product
- 8) Accessibility – the product should be tailored to the needs of different users, e.g. people with disabilities
- 9) Universality – the software product should accommodate a diversity of users
- 10) Usefulness – the product should help users to solve real problems.

The usability sub-factors are shown in table 4.2. The relationships between the usability sub-factors (criteria) and main factors are represented in table 4.3. Depending on the evaluation, different weights might be attached to various criteria that contribute to a particular factor.

Criterion	Description
Time behavior	Capability to consume appropriate task time when performing its function.
Resource utilization	Capability to use appropriate amounts and types of resources when the software performs its function.
Attractiveness	Capability to be attractive to the user (e.g. design).
Likeability	Whether the user likes the product.
Flexibility	Whether the user interface can be tailored to suit personal preferences.
Minimal Action	Capability to help users to achieve their tasks in a minimal number of steps.
Minimal Memory Load	Whether user can keep a minimal amount of information in mind in order to achieve a specified task.
Operability	Amount of effort necessary to operate and control a software product.
User Guidance	Degree to which the user interface provides context-sensitive help and meaningful feedback when errors occur.
Consistency	Degree of uniformity of UI elements and whether they offer a meaningful metaphors to users.
Self-Descriptiveness	Capability of the product to convey its purpose and give clear assistance in its operation.
Feedback	Ability to respond to user inputs or events in a meaningful way.
Accuracy	Capability to provide correct results or effects.
Completeness	Whether the user can complete the specified task.
Fault Tolerance	Capability to maintain a specified level of performance given faults or errors.

Criterion	Description
Resource Safety	Whether resources (including people) are handled without any hazard.
Readability	Ease with which visual content (text dialogs) can be understood.
Controllability	Whether user feels that they are in control of the product.
Navigability	Whether users can move around in the application in an efficient way.
Simplicity	Whether the extraneous is eliminated from the user interface without significant information loss.
Privacy	Whether personal information is appropriately protected.
Security	Whether data is appropriately protected.
Insurance	Concerns the liability of the software product vendors in the case of fraud.
Familiarity	Whether the user interface offers recognizable elements that can be understood by the user.
Load Time	Time required for a Web page to load (fast response to the user).
Appropriateness	Whether visual metaphor in the UI are meaningful.

Table 4.2: Usability criteria in QUIM [SeDoKI05, p.12]

Criterion	Factors									
	Efficiency	Effectiveness	Satisfaction	Productivity	Learnability	Safety	Trustfulness	Accessibility	Universality	Usefulness
Time behavior	+			+						
Resource utilization	+			+						+
Attractiveness			+						+	
Likeability			+							
Flexibility		+	+					+	+	+
Minimal Action	+		+		+			+		
Minimal Memory Load	+		+		+			+	+	+
Operability	+		+				+	+		+
User Guidance			+		+			+	+	
Consistency		+			+	+		+	+	
Self-Descriptiveness					+		+	+	+	
Feedback	+	+							+	+
Accuracy		+				+				+
Completeness		+				+				
Fault Tolerance						+	+			+

Factors										
Criterion	Efficiency	Effectiveness	Satisfaction	Productivity	Learnability	Safety	Trustfulness	Accessibility	Universality	Usefulness
Resource Safety						+				
Readability								+	+	
Controllability							+	+	+	+
Navigability	+	+					+	+	+	
Simplicity					+			+	+	
Privacy							+		+	+
Security						+	+			+
Insurance						+	+			
Familiarity					+		+			
Load Time	+			+					+	+
Appropriateness							+	+	+	+

Table 4.3: Relationship between factors and criteria (subfactors) in QUIM [SeDoKI05, p. 13]

Device Independence Authoring Challenges

In addition to the general set of quality attributes, frameworks for device-independent content authoring should fulfill certain requirements that will help authors in building Web applications that are accessible to users via different access mechanisms. These requirements are also called authoring challenges for device independence and are described in the World Wide Web Consortium Group Note [W3C03]. The challenges refer to authors in different roles: interface designers (layout designers, stylistic designers, navigation designers, and interaction designers), content creators, and business logic creators. Interaction designers are responsible for the design of a site in terms of its graphical look, layout, navigation, and interaction possibilities. Content creators prepare Web contents in the form of text, images, audio, and video resources. Business logic creators are in charge of creating application logic for application sites such as on-line shops or Web interfaces to various enterprise applications. Project managers are responsible for supervising the work and maintaining its quality and efficiency.

The specified authoring challenges refer to different requirements for device-independent approaches such as the comprehensiveness of scope, smooth extensibility of applications, simplicity, abstraction, variability of delivery context, author-specified variability (with regard to content, media, navigation, layout), client-side processing, use of existing standards, context-awareness of applications, and affordability.

Authors participating in the adaptation process are interested in different aspects of adaptation frameworks. For content managers, easy-to-use update and versioning mechanisms are important. Interface designers are mainly interested in attractive, graphical looks of Web pages, consistent navigational models and easy-to-manage adaptation mechanisms for navigation, structure and layout

transformations. Business logic designers expect that frameworks will separate business logic aspects from presentation aspects. Project managers focus on economical aspects such as the possibility to decrease the implementation and maintenance costs or to reduce time-to-market. The overall goal of people participating in the adaptation process is to design and construct extensible, changeable services that will provide satisfactory user experience for different access mechanisms.

Since device-independent content generation approaches can be evaluated with the help of these factors, they are named and briefly explained in table 4.4. The numbers in the table correspond to the numbers of authoring challenges (e.g. DIAC-3.1, etc.) in the W3C specification.

Authoring Challenge	Description
DIAC-3.1: Application scope	Device-independent authoring techniques should be able to support all types of Web applications for all devices.
DIAC:I-3.2: Extensible capabilities	Authoring techniques should be extensible and allow extending these techniques to new technologies.
DIAC:I-3.3: Affordability	The development of device-independent applications should not require excessive effort or high investments.
DIAC:I-3.4: Simplicity	Simple applications should be developed without great effort; the complexity of development should scale smoothly with the complexity of applications.
DIAC:I-3.5: Navigation support	Various navigation methods should be supported and they should be implemented with minimal effort.
DIAC: I-3.6: Organization	The amount of content displayed on various devices with different presentation techniques should vary.
DIAC: I-3.7: Media variety	On different devices different media types should be supported if necessary (e.g. audio, video, etc.).
DIAC: I-3.8: Interaction abstraction.	Interactions with the users should be specified in an abstract form and not defined separately for each device type.
DIAC:I-3.9: Interaction abstraction scope	Data entry, data selection, control functions, and navigation should be covered in an abstract interaction definition.
DIAC:I-3.10: Interaction organization	The variability of displayed control (depending on a device) should be extended to interaction elements.
DIAC:I-3.11: Simple content	Authoring techniques should enable the use of the simplest forms of content.
DIAC:I-3.12: Text content variety	Different versions of text should be supported and displayed depending on the device.
DIAC:I-3.13: Media resource variety	Various media formats should be supported according to the capabilities of devices.
DIAC:I-3.14: Media resource specification	Authors should be able to define lists of media types for different delivery contexts.
DIAC:I-3.15: Media resource selection	Authoring techniques should enable the ability to select a resource from a list of possibilities.
DIAC:I-3.16: Media resource conversion	Conversion from one media type to another one should be possible.
DIAC:I-3.17: Author control of media resource use	Authors should be able to execute control over forms of media resources used.

Authoring Challenge	Description
DIAC:I-3.18: Application data integration	Author should be able to deliver various presentations of application data depending on the display size and storage capabilities.
DIAC:I-3.19: Integration of device-independent content	It should be possible to include external, device-independent content into existing application.
DIAC:I-3.20: Integration of device-dependent content	It should be possible to integrate external, device-dependent content into existing content.
DIAC:I-3.21: Integration of non-presentation markup	Markup that does not include presentation definition should not be prevented from inclusion.
DIAC:I-3.22: Dynamic media resource specification	The used media resources should be specified not only in a static but also in a dynamic way.
DIAC:I-3.23: Use of client-side functions	Authoring techniques should include the possibility of using client-side functions (e.g. for processing).
DIAC:I-3.24: Abstraction of client-side functions	Authoring techniques should enable the use of client-side functions without the necessity to code this explicitly for various devices.
DIAC:I-3.25: Independence from client-side functions	Server-side functions should be supported for harmonized user experience in the absence of appropriate client-side functions.
DIAC:I-3.26: Explicit use of client-side functions	Authoring techniques should enable explicit definition of client-side functions if necessary.
DIAC:I-3.27: Rich media interaction	Authoring techniques should provide abstraction of rich media interaction without specifying the device characteristics explicitly.
DIAC:I-3.28: Range of complexity	Simple and complex applications should be supported by one technique.
DIAC:I-3.29: Scalability of complexity	Increasing application complexity should imply rising complexity of the authoring technique.
DIAC:I-3.30: Aggregation	Aggregation of presentation units should be supported.
DIAC:I-3.31: Fragmentation	Decomposition of resources should be possible.
DIAC:I-3.32: Abstract relationships	Relationships between authoring units should be possibly abstract to derive navigation structures without effort.
DIAC:I-3.33: Existing applications	Authoring techniques should provide support for including parts of existing applications during the design or at runtime.
DIAC:I-4.1: Device capabilities	Wide range of device features should be supported.
DIAC:I-4.2: Capability abstraction	Authoring techniques should be able to specify desired presentations using abstract definitions of device capabilities.
DIAC:I-4.3: Layout	Different spatial and temporal layouts should be supported for different delivery contexts.
DIAC:I-4.4: Presentation usability	Authoring techniques should help in creating presentations with usability appropriate for a device.
DIAC:I-5.1: Presentation personalization	Personalization of presentation according to user preferences should be possible.
DIAC:I-5.2: Modality	Device input modalities should be considered in the adaptation process.
DIAC:I-6.1: Presentation markup	Presentation should be based on existing markups such as XHTML, SVG, and SMIL.
DIAC:I-6.2: Interaction markup	Interaction markup should be based on XForms standard.

Authoring Challenge	Description
DIAC:I-6.3: Non-presentation markup	Markup not associated with the presentation or interaction should be based on XML.
DIAC:I-6.4: Transformation of markup	Transformation should be based on XML and XSLT.
DIAC:I-6.5: Minimization of effort	Support for a variety of devices should be possible without excessive effort.
DIAC:I-6.6: Abstraction of device knowledge	Detailed knowledge about the target devices should not be required.
DIAC:I-6.7: Functional presentation	Functional user experience should be possible with minimal effort.
DIAC:I-6.8: Customized presentation	Customization of user experiences for particular devices should be possible.
DIAC:I-6.9: Exploitation of device capabilities	Customizations should help in exploiting device-specific capabilities.
DIAC:I-6.10: Scalability of effort	The author should be able to decide how much effort he/she would like to invest in the customization process.
DIAC:I-6.11: Future device capabilities	Authoring techniques should be extensible to support future device capabilities.
DIAC:I-6.12: Reuse	Reuse of materials should be possible.
DIAC:I-6.13: Separation	Device-independent material should be separated from device-dependent material.
DIAC:I-6.14: Reuse of functional presentation	Existing user experience should be accessible to future devices without considerable effort.

Table 4.4: Device Independence Authoring Challenges (DIACs) [W3C03]

4.4 Design of mobile applications

Appropriate design of mobile applications is a very important factor in their adaptation processes. Mobile devices possess limited capabilities but offer various advantages such as portability and instant access to time-critical information. In mobile communication, applications are used by people being in motion or performing additional tasks. Users want to access information easily, by clicking only on a few buttons. This approach is called “information hunting” as compared to traditional “Internet surfing” accomplished with the help of desktop computers [Weis02]. Mobile applications are usually presented on relatively small screens. They are supposed to provide short response times and display only small amounts of relevant data. Researchers point out that the information structure and screen size can considerably affect the navigation behavior and the users’ perception of mobile Internet. User’s behavior can be further influenced by the complexity of a task [ChaKi04].

Particularly challenging design issues in the process of mobile applications development include [cf. Björ+02; KaRo03; Wall+02]:

- visualization of information and appropriate navigation mechanism (due to the limited display)
- limited interaction possibilities (e.g. one-handed use)

- unpredictable context of use (unstable environment in comparison to desktop-based solutions)
- access speed (possibility to access information quickly)
- development costs.

Human-computer interaction designers identify the following five main challenges in the design of mobile interfaces [cf. DuBr02]:

- designing for mobility - the working environment of mobile users is quite chaotic and it changes as the user moves
- designing for widespread population - users do not have any specific training concerning mobile devices, mobile handsets are used by different groups of people (e.g. young and older people, well-educated specialists and average citizens, without specific technical knowledge)
- designing for limited input/output facilities - input and output quality will remain poor even though the screen sizes will improve
- designing for incomplete and varying context information - some contextual information can be easily captured (e.g. location) while other information still remains difficult to specify
- designing for users multitasking at levels unfamiliar to most desktop users - multitasking and support for task interruption is very important for the design of desktop interfaces, in mobile context the frequency of such interruptions dramatically increases.

Content and navigation design guidelines for mobile devices include the following [cf. Buch+01; JoBuTh02; Jone+99; Kaas+00; Tsch+02; WeGr02]:

- present the most important content at the top of a page, keep the content compact (short pages are recommended)
- simplify page layout - avoid elements that do not add any value to the existing content
- use simple text elements and styles
- provide short but descriptive page titles
- take into account the acceptable size of a document
- use compact link names
- design clear forms - forms should be possibly short, with the possibility to cancel the form filling process and to go back
- provide informative, small and simple graphics
- minimize the number of steps in the navigation - research showed that users are frustrated if they have to scroll through long lists of options or to fill out complex search forms
- provide selecting options instead of typing - users prefer to choose from a default list using select lists, checkboxes, or radio buttons rather than typing
- keep navigation consistent throughout the entire service
- design flat menus
- cross-link the pages (provide the "Back" functionality for every page)
- provide confirmations for important actions
- provide intuitive searching mechanisms.

Many research studies emphasize the fact that usability is the most important factor influencing the acceptance of mobile solutions. The Microsoft Usability Guidelines (MUG) [Keek97] name five crucial attributes of site design intended to increase their usability:

- content - refers to the informational and transactional capability. Important subcategories of content include relevance (content should be of interest to the users), media use (media should be used appropriately and effectively to communicate the content), depth and breadth (appropriate content level), and currency (timeliness of content). A useful site should successfully deliver timely information or entertainment in the right way and at the right level, or improve the way users accomplish a task
- ease of use - refers to the ease of interacting with a page. A site that is easy to use should be straightforward to understand, well-structured, and easy to navigate around. It should also deliver clear and understandable results regarding its progress of use (e.g. the user should understand the navigation structure)
- made-for-the-medium - refers to the extent to which a page can be tailored to fit specific user needs. Particularly important are personalization of sites and their refinement that should reflect past and current trends (e.g. by maintaining an archive)
- emotion - refers to the emotional reactions a site can raise. Not all sites should lead to emotional reactions but emotions are particularly important for entertainment services, e.g. games
- promotion - refers to the way a site is promoted on the Internet or through other media.

The research of Venkatesh, Ramesh, and Massey [VeRaMa03] focused on the evaluation of the usability of wireless Web sites. They found out that content (particularly its relevance) was essential to users regardless of whether a site was wired or wireless. The ease of use and made-for-the-medium criteria were considerably more significant in wireless contexts, largely due to their subcategories structure and personalization. Promotion and emotion was not of great importance for wireless pages. Therefore, the results of this study suggest that relevant and easy to navigate content, as well as the medium's ability to personalize content are indispensable in creating wireless interfaces that will be adopted by wide audience.

In 2006, the World Wide Web Consortium published the Mobile Web Best Practices recommendation. This document aims at improving the "mobile Web experience" [W3C06a] and should help to realize the concept of "one Web" that assumes that the same services and information should be available on all device types. The best practices refer to the overall behavior of mobile pages (e.g. exploitation of delivery context), navigation and links (e.g. short URIs, use of access keys), page layout and content (e.g. scrolling, page size limits), page definition (e.g. tables, frames, etc.), and user input (labels for controls, minimal number of keystrokes). Apart from providing recommendations for the development of mobile applications, they also propose methods to test the realization of the recommended practices. Table 4.5 summarizes the most important practices outlined in the W3C specification.

Best practice	Description
Thematic consistency	Ensure that content provided by accessing a URI yields a thematically coherent experience when accessed from different devices (the realization of the “one Web” concept).
Capabilities	Exploit delivery context to provide an enhanced user experience.
Deficiencies	Take reasonable steps to work around deficient implementations (e.g. different rendering of markup elements by various browsers).
Testing	Carry out testing on actual devices and emulators.
URIs	Keep the URIs of site entry points short to avoid unnecessary typing.
Navigation bar	Provide only minimal navigation at the top of a page.
Balance	Take into account the trade-off between having too many links on a page and asking the user to follow too many links to reach what they are looking for.
Navigation	Provide consistent navigation mechanisms (similar on all devices to facilitate orientation).
Access keys	Assign access keys to links in navigational menus and frequently accessed functionality.
Link targets, link target format	Clearly identify the target of each link (size of data, content type). Note the target file's format unless you know the device supports it.
Image maps	Do not use image maps unless you know the device supports them effectively.
Pop-ups	Avoid using pop-ups.
Auto refresh	Do not create periodically auto-refreshing pages, unless you have informed the user and provided a means of stopping it.
Redirection	Do not use markup to redirect to other pages automatically.
External resources	Keep the number of externally linked resources to a minimum as it requires a separate request across the network.
Suitable, limited content	Provide context-sensitive content. Limit content to what the user has requested (e.g. avoid advertising).
Clarity	Use clear and simple language (discursive style is not recommended for texts).
Page size, page size limit	Divide pages into usable but limited size portions. Ensure that the overall size of a page is appropriate to the memory limitations of the device.
Scrolling	Limit scrolling to one direction, unless secondary scrolling cannot be avoided.
Central meaning	Ensure that material that is central to the meaning of a page precedes material that is not (relevant content should be displayed first).
Spacing	Do not use graphics for spacing.
Large graphics	Do not use images that cannot be rendered by the device. Avoid large or high resolution images.
Use of color, contrast	Ensure that information conveyed with color is also available without color. Ensure that foreground and background color combinations provide sufficient contrast.
Background images readability	Make sure that content remains readable on the device when background images were used.
Page title	Provide a short but descriptive page title.
Frames	Do not use frames.

Best practice	Description
Structure	Use features of the markup language to indicate logical document structure (headings and sub-headings).
Tables, nested tables	Do not use tables unless the device is known to support them. Do not use nested tables.
Layout with tables	Do not use tables for layout.
Alternatives to tables	Where possible, use an alternative to tabular presentation.
Non-textual alternatives	Provide a text equivalent for every non-textual element.
Scripts/objects	Do not rely on embedded objects or scripts.
Valid markups	Create documents that validate to published formal grammars.
Images size, resizing	Specify the size of images in markup, if they have an intrinsic size. Resize images at the server, if they have an intrinsic size.
Measures	Do not use pixel measures and absolute units in markup language attribute values and style sheet property values.
Stele sheets	Use style sheets to control layout and presentation, unless the device is known not to support them.
Stele sheets support Style sheets size	Organize documents so that if necessary they may be read without style sheets. Keep style sheets small.
Efficient markup	Use terse, efficient markup (avoid stylistic effects leading to pretty printing).
Content format support	Send content in a format that is known to be supported by the device. Where possible, send content in a preferred format.
Character encoding	Ensure that content is encoded using a character encoding that is known to be supported by the device. In the response, indicate the character encoding being used.
Error messages	Provide informative error messages and a means of navigating away from an error message back to useful information.
Cookies	Do not rely on cookies being available.
Caching	Provide caching information in HTTP responses.
Fonts	Do not rely on support of font-related styling.
Keystrokes	Keep the number of keystrokes to a minimum.
Avoid free text, default entries for input	Avoid free text entry where possible. Provide pre-selected default values where possible. Specify a default text entry mode, language and/or input format if the device is known to support it.
Tab order	Create a logical order through links, form controls, and objects.
Control labeling and position	Label all form controls appropriately and explicitly associate labels with form controls. Position labels so they lay out properly in relation to the form controls they refer to.

Table 4.5: Mobile Web best practices [W3C06a]

5

Technologies for device independence

Displaying Web information on small devices in a device-independent way is a field of ongoing development and research. Different attempts to address this problem have been undertaken since the mid-nineties. Previous research in the area of adaptation frameworks can be sorted into three broad categories [cf. BiSC97; Butl+01; Butl+02]: client-side adaptation, server-side adaptation, and intermediate adaptation. Researchers and practitioners use four categories of methods for displaying Web pages on wireless clients: manual authoring, scaling, transducing, and transforming [cf. Schi+01; Schi+02]. In one framework supporting various devices different methods can be applied. Usually, scaling is combined with some sort of transcoding or transforming [cf. Björ+99; ChMaZh03].

This chapter categorizes the technologies for device independence and is composed as follows: section 5.1 introduces the methods for device-independent content adaptation. It briefly outlines the state of the art in the area of adaptation techniques and presents some relevant implementations. At the end of this section, a comparison between manual authoring, scaling, transducing, and transforming is provided. Section 5.2 focuses on a general description of adaptation approaches and highlights their advantages and shortcomings.

5.1 Methods for content adaptation

Numerous mobile browsers do not offer any support for the language that contributed to the worldwide acceptance of the Internet and that is still the main markup for the Web - HTML. Therefore, automated content adaptation often relies on translating HTML sites into markup languages that are compatible with wireless browsers such as WML, XHTML, or cHTML. Multimedia content has to be converted to displayable formats or replaced by textual descriptions. For example, on simple WAP phones JPEG images have to be transformed into WBMP graphics. Furthermore, the layout of pages, the entire navigational structure and the amount of information presented on one screen, has to be changed if rich hypermedia content targets small displays. Alternatively, the content can be created in a device-independent language and can be dynamically converted to an end-format or it can be embedded into Java ME forms. Tailoring traditional Web sites to wireless devices can be achieved by manual re-authoring of existing pages or by applying automated approaches such as scaling, transforming, or transcoding.

5.1.1 Porting the Web to mobile devices - browser-based rendering

Before investing money and time in the development of adaptation approaches, it is worth to examine mobile browsers with regard to their abilities to render traditional Web sites. The most popular browsers for mobile devices (also called microbrowsers) are as follows:

- Pocket Internet Explorer⁵³ - is the default browser on Windows CE, currently in version 4. It works with Windows Pocket PC and Handheld editions. Pocket Internet Explorer supports HTML 4.0, DHTML (IE4 DOM), XML/XSL (data islands), CSS, JScript (ECMA 262), VRML (embedded viewer), and 64-bits encryption (SSL). ActiveX controls can also be downloaded.
- NetFront⁵⁴ - is an embedded browser that can be found on wireless phones, PDAs, set top TV boxes, and even on the Sony PlayStation Portable (PSP). The engine of this browser is based on this one used in Palm's Blazer. NetFront supports HTML 4.01, XHTML, cHTML, WML 1.3, CSS 1 and 2, JavaScript 1.5, DOM 2, and Dynamic HTML. The newest version of NetFront (3.4) comes with SMIL 2.1 and SVG 1.2 support, RSS viewer libraries, and AJAX. It offers three different ways to render a page and is mostly used by Sanyo in USA and DoCoMo in Japan. NetFront is also implemented on Sony Ericsson's high-end feature phones (W810i, W550i, W900i). The browser requires 4 MB of memory and consumes between 5 and 16 MB RAM for page rendering.
- Minimo⁵⁵ - is an open-source browser using the same engine as Mozilla and Firefox. The most important problem of this browser is the required memory amount – so far Minimo development has centered around devices with 64 MB of RAM such as Hewlett-Packard's iPAQ. Minimo supports spatial navigation that allows changing the focus on a page and possesses text zooming capabilities to increase the size of the read text. Additionally, it supports Small Screen Rendering based on CSS and adjusts the pages by changing image sizes, fonts, and layouts.
- Series 60 Browser⁵⁶ - is a new browser used by Nokia for the third edition of its Series 60 handsets. The browser is based on the WebCore and JavaScriptCore components of Apple's Safari Web Kit as well as on KHTML and KJS from KDE's Konqueror open source project. S60 browser supports DHTML, AJAX applications, HTML, XHTML 1.0, DOM, CSS and SVG-Tiny, ECMAScript, and Netscape style plug-ins such as Flash Lite and audio. S60 offers minimaps that allow to see the page as a small thumbnail and to navigate through it very fast on a large page. The browser also supports RSS and have the "Search as you type" functionality implemented (autocompletion of text).
- Opera Mobile⁵⁷ and Opera Mini⁵⁸ (cf. sections 6.1.1 and 6.2.8 for more details) - Opera Mobile is the default browser on a large number of mobile devices. It supports CSS, JavaScript, AJAX, WML and has a Small Screen Rendering mode that adjusts standard Web pages to the features of mobile devices. Opera Mobile needs about 4-5 MBs of RAM to start the browser, therefore it mostly runs on PDAs, smartphones, or high-end feature phones. Opera Mini is a Java-based browser and can only run on Java-enabled phones. It pre-processes the sites on a remote server and sends them to a mobile device instead of using CSS for better rendering.

⁵³ <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceinternet5/html/wce50oripocketinternetexplorer.asp>.

⁵⁴ <http://www.access.co.jp/english/products/nf.html>.

⁵⁵ <http://www.mozilla.org/projects/minimo>.

⁵⁶ <http://opensource.nokia.com/projects/S60browser/>.

⁵⁷ <http://www.opera.com/products/mobile/products/>.

⁵⁸ <http://www.opera.com/products/mobile/operamini/phones/>.

- OpenWave⁵⁹ - is a very popular browser for low-end handsets. It can be found, for example, on Samsung, Siemens, Sharp, LG and Sprint phones. Openwave's browser is able to run on handsets with only 512 KB of allocated memory. In its newest version (Mercury Edition), the browser supports WML, XHTML MP, CSS, AJAX, i-(X)HTML, custom KDDI and J-Phone tags, JavaScript, and downloading of MIDlets. Although the browser needs little RAM to start up, some phone manufacturers limit the amount of memory assigned to it and complex sites cannot be displayed (e.g. some manufacturers give it only 700 KB of RAM and the cnn.com site requires 4 MB to be rendered).
- Teleca's Obigo⁶⁰ - is the biggest competitor for Openwave. It is designed for low-end phones and needs less than 600 KB of memory to start. The browser supports HTML 4.01, CSS 2.1, Javascript 1.5, DOM2, SVG Tiny, XHTML, WML and it incorporates a "thumbnail view". Obigo offers messaging capabilities and a media player that can be found on some Samsung and LG phones.

Mobile browsers are able to display simple Web pages more or less correctly. Complex designs (e.g. layouts based on nested tables, frames) and pages overloaded with graphics are usually turned into gibberish by the browsers. Figure 5.1 shows two Web sites rendered by the most popular browsers: the cnn.com page and Microsoft's MSDN page. Most browsers had problems with displaying frames on a small screen what led to the need for horizontal and vertical scrolling. Some browsers were unable to display images (e.g. Minimo, Openwave) and the produced output was not user-friendly. For example, in order to benefit from the zooming functionality of S60 browser, the user has to know the desktop version of a page, otherwise he/she has to guess which part of the page should be scaled because the mobile page is so small. All of the browsers do not paginate Web sites, they simply display them as one long page.

Relying on available microbrowsers is partially consistent with Device Independence principles because the same URI may be used to access a page from a desktop browser and a microbrowser. "One suits it all" version is produced for all device types and therefore no additional investments are necessary. However, the available microbrowsers do not address the content-, context- and feature-specific needs of mobile users. Decreasing font sizes and the introduction of new navigation structures relying on vertical scrolling cannot be considered as a realization of the dream of mobile browsing. The usability of pages rendered by microbrowsers is not at a satisfactory level since they are not easy to use and were not made specifically for mobile devices but are only slightly reformatted. Further disadvantages of this solution include different, browser-specific rendering capabilities and the need to rewrite existing Web pages according to the HTML guidelines for mobile access.

⁵⁹ http://www.openwave.com/us/products/device_products/mobile_browser/index.htm.

⁶⁰ <http://www.obigo.com/>.

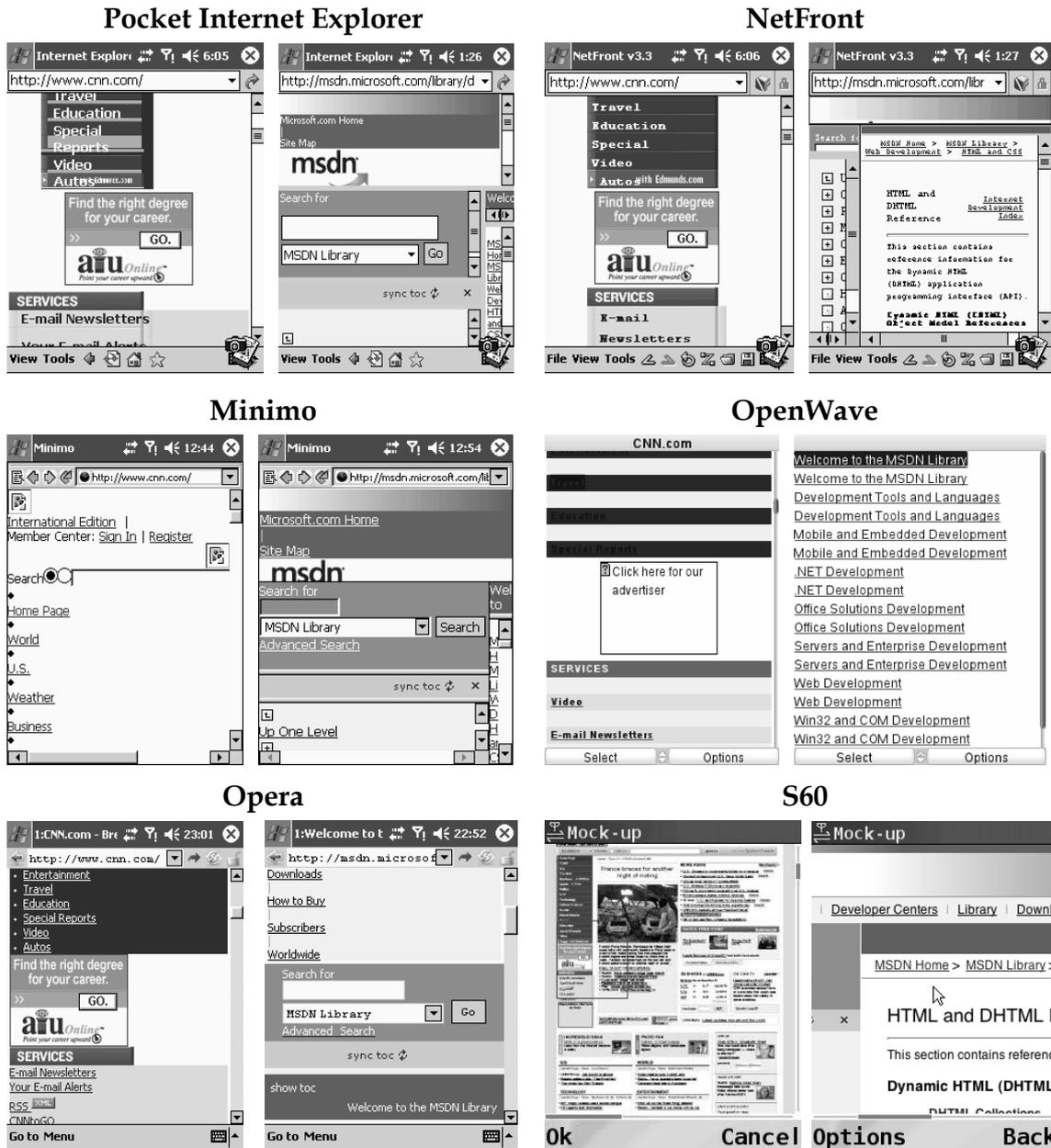


Figure 5.1: Exemplary HTML pages rendered by microbrowsers [http://www.howtocreate.co.uk/operaStuff/devices]

5.1.2 Manual authoring and the “versionitis” problem

Manual authoring is a very work-intensive method consisting of the duplication of existing information and creation of separate content that fits particular devices or classes of devices [Kaas+00]. This approach leads, at least theoretically, to well-designed pages with good layout and navigation possibilities, but it is only feasible if the number of pages is small or enterprises can afford high development and maintenance costs. Manual authoring is also directly associated with the “versionitis” problem – creation of different versions of the same page with proprietary, idiosyncratic designs for different operators supporting various mobile standards.

Many large enterprises such as Volkswagen AG, Bayerische Hypo- und Vereinsbank AG, or IBM created separate sets of pages, viewable only on mobile browsers⁶¹. Their WAP pages present the most important news and products and do not aim at the replication of existing, frequently changing Web sites. The maximal number of mobile sites offered by an enterprise (according to the index created in the “*Wireless Usability Software Agent*” (*WUSA*) project [CeKuNo03; CeKuNo04]) amounts to 501, whereby firms which are apparently not convinced about the potential of m-commerce developed only 1 page⁶².

Although manual authoring should theoretically provide the best user experience, a recent study on 77506 WML pages, conducted by the authors of the *WUSA*, revealed that manually re-authored pages are not designed and developed carefully [CeKuNo04]. The study showed that 19 percent of the examined pages were faulty. Some types of errors were particularly frequent. 31 percent of the incorrect sites referred to dead links or links that could not be displayed in mobile browsers. 27 percent of the erroneous sites could not be shown because they exceeded the allowed deck size of 1400 octets. 32 percent of the pages were not displayed due to various violations of the WML-DTD specification (syntax errors) [CeKuNo04, pp. 61-62].

The *Wireless Usability Software Agent* brought some important and fearsome insights into the current state of the manual development of WAP pages. It disclosed, for example, that the average number of lines on one card amounts to 8.3 and the maximal number of lines is as much as 498. The detailed examination of some chosen mobile services confirmed the fact that manually developed pages often ignore device independence principles and are designed only for a limited group of owners of modern phones with big, color displays.

Common problems encountered during mobile browsing of manually designed pages can be categorized into the following groups:

- a) difficulties related to content - mobile content is usually reduced to the most important snippets of information for which the user has to wait around 30 seconds to download. Alternatively, mobile sites are overloaded with information and the user breaks the connection, because he/she is not able to find the needed data. Some browsers have certain limits concerning the maximal page length – usually it cannot be larger than 5kB. If a page is bigger, the user is confronted with the “413” error. Browsers that possess such limitations can be, for example, found on Motorola’s RAZR phones. The images are often too large to be displayed on a mobile page or they are so tiny that the user cannot extract any information from them. Mobile financial services frequently place charts displaying the development of stock prices⁶³. However, the user is not able to benefit from this form of presentation because he/she is usually able to see only some sort of line and not the relevant details of the chart like prices, dates, etc.

⁶¹ Volkswagen authored manually 63 WAP pages (<http://wap.volkswagen.de>), Bayerische Hypo- und Vereinsbank 495 pages (<http://wap.hypovereinsbank.de>), and IBM 492 pages (<http://wireless.ibm.com/de>).

⁶² The project’s homepage is available at <http://projekt.wifo.uni-mannheim.de/wusa/index>.

⁶³ Cf. <http://wap.hypovereinsbank.de>.

- b) formatting problems - even though XHTML-MP phones provide support for basic formatting (alignment, font size, colors, tables, and standard images) and are able to display the basic version of WAP CSS style sheets, many mobile pages are not formatted at all and some elements are just dispersed all over the page.
- c) problems with navigation - mobile pages are sometimes too complex and the user is not able to find any navigational help or even perform such a simple operation as going to the next or previous page.
- d) developers' faults - problems resulting from errors forgotten or ignored by developers such as dead links, only partially implemented functionality, pages producing errors, etc.

Figure 5.2 shows some examples of improperly designed/developed mobile pages. Figure 5.2a demonstrates a mobile page of Otto Versand service⁶⁴ displaying offered products from the category "Bargains". The service possesses only a German version of this mobile site. The user is supposed to scroll down for a long time until he/she can finally reach some links that will allow him/her to go back to the homepage or to find more products. The authors of this page tried to squeeze as much information as possible on one enormous mobile page with a lot of links. Users will probably have difficulties reading this page on a smartphone, but it should not be forgotten that the page was also designed for mobile phones with small screens such as the one presented in figure 5.2b. It is neither appropriate nor user-friendly to assume that all potential clients have recently switched from their old devices to new and powerful phones with big displays and to test the pages only on such phones. Figures 5.2c and 5.2d show some examples of mobile pages with errors. In figure 5.2c the input was not recognized and the user gets an empty list of results with pagination possibilities. Figure 5.2d displays an error message that a normal user is not even able to understand.

Although billion of Web pages were created so far, mobile Internet is still in its infancy considering the number and quality of available sites or portals. Some pages and catalogues include lists of mobile sites (e.g. <http://www.evmo.com/MobileWeb/MobileWeb.aspx> or <http://blogs.msdn.com/windows/mobile/articles/216788.aspx>) and Google engine performs even a special Web crawl to create a separate index of WML/XHTML pages⁶⁵. However, the results of this crawl leave a lot to be desired. Even though the choice is small, many of the manually authored pages produce errors, consist of one site with a bunch of dead links, or can be viewed only on particular types of devices.

One could expect that the situation is different in Japan that is perceived as the early adopter of mobile computing and manufacturer of the most advanced mobile devices. Surprisingly, the "versionitis" problem is at least as acute there as in Europe, although mobile Web pages and mobile browsing are far more popular in this country [cf. Bov05]. For example, in order to access the Japanese version of Toshiba's Web site from a desktop computer, users just have to type the <http://www.toshiba.co.jp/> address in the browser's address bar. For mobile users, three carrier-specific versions were prepared:

⁶⁴ Cf. <http://wap.otto.de>.

⁶⁵ Cf. http://www.google.com/mobile/mobile_search.html.

<http://www.toshiba.co.jp/keitai/i/> (i-mode) for NTT DoCoMo subscribers, <http://www.toshiba.co.jp/keitai/j/> (*Vodafone live!*, formerly *J-Sky*) for Vodafone KK subscribers, and <http://www.toshiba.co.jp/keitai/ez/> (*EZweb*) for AU subscribers. These mobile versions are good examples of “tag soups” (cf. section 2.2.1) and include a lot of proprietary markup elements. The EZweb version is written in HDML that is considered to be an obsolete language.

Each mobile provider in Japan uses a different standard, usually with some proprietary extensions. NTT DoCoMo supports i-HTML and i-XHTML and guarantees backward compatibility with its previous standards. When in 2003 J-Phone became Vodafone KK, it started to support XHTML Basic, XHTML-MP, and CSS 2. Before this change occurred, J-Sky services supported Mobile Markup Language (MML) that was similar to NTT DoCoMo's cHTML with some J-Sky specific extensions. For backward compatibility reasons, Vodafone KK cell phones also render a number of non-standard proprietary tags and attributes. There is no cross-compatibility with the emoji supported on NTT DoCoMo's or AU handsets. At the beginning, KDDI's AU carrier supported Handheld Device Markup Language (HDML), but its new devices offer support for WML, cHTML, XHTML Basic, XHTML-MP and WAP CSS. Similarly to other providers, AU browsers can interpret proprietary markup elements and AU emoji follow a proprietary mapping system.

Following the approaches of mobile providers, most mobile Web pages come in three flavors (i-mode, Vodafone live!, EZweb). Examples include the pages of Asahi.com, Tokyo Hands, Yahoo! Japan, FujiTV, and Sony Japan⁶⁶. Some of these pages check for the browser's type and cannot be accessed with a desktop browser. For a number of sites, the same URL is given for all versions and the user is redirected to the correct version depending on his/her browser. Several Web services are available in more versions than the „obligatory“ three. For example, the Japan Airlines website⁶⁷ has seven versions: classic, i-mode, Vodafone live!, EZweb, Air-Edge,⁶⁸ L-mode⁶⁹ and PDA. It is important to note that although Web standards are ignored in Japan and the available pages are not compliant with the W3C Device Independence Principles, mobile browsing became widely accepted.

With regard to Device Independence Principles, manual authoring enables device-independent access if appropriate pages are prepared and tested on various categories of devices. However, in most cases the content is re-authored for wireless clients supporting either WML or XHTML. Users have to enter different URLs to see Web pages in desktop browsers and mobile browsers; device-independent Web page identifiers are therefore missing. The offered functionality is similar to the one available for desktop computers, but usually limited in scope since Web pages are duplicated only partially.

⁶⁶ <http://www.asahi.com/>, <http://www.tokyu-hands.co.jp/>, <http://www.yahoo.co.jp/>, <http://www.fujitv.co.jp/>, <http://www.sony.co.jp/>.

⁶⁷ Cf. <http://www.jal.co.jp/mobilephone/url.html> for a list of versions.

⁶⁸ Air-Edge is a wireless data transmission service that can be used from desktop computers and PDAs. Air-Edge also started offering cell-phone services. More information can be found at: http://www.ddipocket.co.jp/p_s/products/airh_phone/.

⁶⁹ L-mode is a service similar to i-mode. The user can send e-mails and search for information from a fixed phone or fax. More information on L-mode is available at http://www.ntt-east.co.jp/Lmode_e/.



Figure 5.2: Exemplary mobile pages: a) Otto Versand mobile service displayed on a phone with a big screen (176x208 px), supporting colors and images (Nokia 6600) b) The same service displayed on Nokia 6100 with a smaller screen (128x128 px) c) Mobile weather service shown on Sony Ericsson T610, d) Stock quotes from the Swisquote service displayed on Siemens SX1

5.1.3 Scaling

Scaling is one of the oldest methods for content adaptation on mobile devices. It replaced panning (traditionally used in desktop applications), because this method was found to perform poorly for completing tasks on wireless devices [GuFe04]. Scaling allows to increase and to decrease pages in a viewer, using some kind of a “fit-to-screen” feature, or alternating the portion of a page displayed at any given time (e.g. by using the zooming functionality). It is recommended to use scaling on high-resolution color displays (e.g. Pocket PC) equipped with browsers with vertical and horizontal scrolling capability. Scaling reduces the readability of a site and makes the interaction and navigation more difficult. This approach usually suffers from excessive scrolling requirements, especially on small mobile phones with a limited number of displayable text lines [BuMoPa00]. With the course of time, various innovative scaling methods for mobile handhelds emerged. Recent research mainly concentrates on the so-called focus and context techniques and on speed-dependent automatic zooming.

Zooming

Zooming was already used for viewing information at multiple scales on different devices (PDAs, high-end graphics workstations, set-top boxes) by Pad++ in 1994 [BeHo94; BeMe98]. The authors of Pad++, written in C++ and Tcl/Tk, were trying to provide effective access to large amounts of textual and graphical information on various displays. Data were presented in the spatial format that can be effectively processed by the human perceptual system. Appropriate portions of data could be zoomed out and in. They also experimented with hypertext. If the user selected a hyperlink, the linked page was loaded into a small, separate window, on which the focus was put. The window could then be zoomed in or out and the user could go back to the parent window [BeHo94, p. 20].

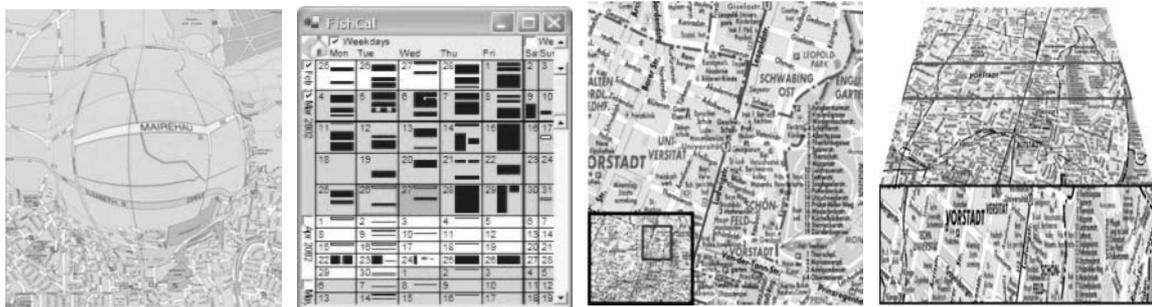
A novel technique proposed by Microsoft that is called “collapse-to-zoom”, tries to hide the irrelevant content such as columns containing menus, archive materials, or advertising. The collapsed fragments can be expanded if necessary, and the user can zoom into important fragments of pages [cf. Baud+04]. However, the main problem with zoomable interfaces is the need for explicit control and coordination of zooming and panning. The users can quickly get lost if the amount of information is large or if the screen is small [JuFu98].

Focus and context techniques

To increase the amount of presented information and to use the space of a mobile screen effectively, specialized focus and context techniques were proposed [cf. GuFe04; KaRoSc04; RaJeSc01; StZh00; Wils+05]. These techniques combine a focus view that presents details of a layout fragment and a context view that shows the whole set of data for overview. Focus and context methods are based on the assumption that a user loses interest in particular parts of the layout with the increasing distance from the focus. The context view can be scaled using uniform-scaling, belt-based scaling, and non-uniform scaling [KaRoSc04].

In uniform scaling, the whole image is divided into a focus and a context belt. The context belt is furthermore split into non-overlapping fragments and each fragment is affected by uniform scaling. Uniform scaling is not computationally expensive, but causes a visible discontinuity if the user moves to new areas that were not displayed previously. Belt-based scaling is used for more adapted scaling of the page. Context belts are generated in a similar way as in uniform scaling, but the belts near the focus are less scaled than the belts near the borders. Belt-based scaling minimizes the visual discontinuities, but requires more computational power. In non-uniform scaling, a single context belt is applied. The context scaling factor increases with the increasing distance from the focus. This approach offers the best presentation results, but it is rather inapplicable on mobile devices with limited power.

Fisheye view, displayed in figure 5.3a and 5.3b [BeCzRo02; RaJeSc01], is a focus and context method that uses different distortion techniques in order to display bulky fragments of images as context information. Near the focus the distortion is small, while near the image borders strong distortion is applied. The fisheye view performs well for menus and navigating tasks [Bede00; GuFe04]. Figure 5.3b presents, for example, a calendar interface designed for PDAs. The interface applies the fisheye technique. The most important calendar entries are much bigger than those that are not relevant for the user [BeCzRo02]. The users may, however, encounter problems with some interaction tasks like targeting [Gutw02].



a) Fisheye map viewer

b) Fisheye calendar

c) Large focus-display

d) Perspective-overview display

Figure 5.3: Visualization techniques in scaling

Focus and context technique was also applied in the so-called WEB Browser for Small Terminals (WEST) that was designed for handheld devices with limited resources (such as 3Com Palm) [Björ+99; BjRe99]. Flip zooming - a special tile-based focus and context visualization method - was employed as the interaction mechanism. In flip zooming, information is split into sets of sequentially ordered discrete objects (images, fragments of texts, or combinations of both types). The focus is set on one object, placed at the center of the screen; the remaining objects are ordered in the left to right, top to bottom fashion and provide contextual information. The user can focus on any visible object by pointing at it and can use the zooming functionality to increase the object. The maximal number of objects displayed on one screen is limited to seven; otherwise the user may become confused while changing the focus to one of the context objects.

A common difficulty in flip zooming approaches is the fact that if the user zooms out for better orientation, she/he cannot see the details on a small display anymore. If the user is zoomed in, the

context can be lost. Additionally, the user has to switch constantly between the main and the context windows. Flip zooming was also applied for an interface called PowerView which showed useful information, commonly used in offices, such as contact lists, emails, tasks, and meetings [Björ+00].

Overview and detail techniques

Overview and detail methods aim at avoiding distortion [CaMaSc99]. They present focus and context information as two separate images.

In a method named large focus-display technique [KaRoSc04], most of the available screen space is used to show the focus and the rest represents the context (cf. figure 5.3c). The context is a downscaled version of an image on which the focus is marked and which is placed at the bottom of the display. The context fragment hides some of the focus area, but the navigation is faster than in alternative approaches.

The perspective overview-display technique presented in figure 5.3d [KaRaSc04] separates the overview (displayed at the top of a screen) and the details of an image (displayed at the bottom of a screen). The overview is warped to save screen space (the same image displayed normally would be much higher) and the focus is marked on the overview for better navigation. The perspective-overview technique has two disadvantages for mobile devices: the method is computationally intensive and the distorted overview does not cover the whole display.

Speed-dependent automatic zooming

Speed-dependent automatic zooming (SDAZ) is the newest scaling approach [CoLoSa03]. It was introduced in 2000 by Igarashi and Hinckley [IgHi00] and combines rate-based scrolling with automatic zooming for better navigation experience. If the user scrolls faster, the system automatically zooms out, giving him/her the impression of a constant scrolling speed [IgHi00, p.139]. As a result, he/she is able to see the desired fragment of information faster and does not become disoriented by the rapidly changing visual flow.

SDAZ was investigated by Smith and Eslambolchilar on Pocket PCs [EsSm04]. They implemented a document viewer for browsing large files in PDF, PS, and DOC format. These formats were converted to an image file (in the Portable Network Graphics format). The viewer was furthermore instrumented with an accelerometer attached to a serial port and responsible for controlling the velocity of scrolling and with a stylus control used for zooming. According to the authors, the initial user evaluation of SDAZ was positive. This approach could provide an intuitive solution to browsing large documents on middle-sized devices such as PDAs or Palmtops.

Scaling theoretically permits device-independent access to Web pages through the same URLs. In practice most scaling approaches are designed for handheld devices supporting only one markup language. Scaling is applicable to wireless devices with good resolution, i. e. handheld devices. It therefore guarantees support for a limited number of devices. Furthermore, this technique can be used as a supportive technique, for example, for displaying large images (e.g. maps) on mobile screens.

5.1.4 Transcoding

Since manual authoring is not a cost-effective solution and scaling cannot be applied on small devices, techniques that allow automatic device-independent content delivery become more and more popular. Transcoding (sometimes also called transducing) is one of such approaches. The term “transcoding” was first used in the media signal processing industry. A signal in the form of a television program was converted from one format to another, while preserving the content of the program. For example the National Television System Committee (NTSC) standard, used in America and Japan, was transcoded to the Phase Alternating Line (PAL) standard used in the rest of the world [Naga03, p. 12].

In wireless environments, transcoding means that HTML tags are either replaced by tags that are supported by alternative browsers (e.g. mobile browsers), or the whole page is converted to an output that resembles the original site. Transcoding is usually performed by some kind of intermediary called transcoding or transducing proxy [Hori+00; HwSeKi02; Naga03]. The proxy retrieves the content associated with a particular URL, transduces it into appropriate formats, and compresses or converts images to supported image types [ChEI99; ChEIVa00; ChEIVa99].

Mobile Google⁷⁰ is a wireless version of Google search engine that transduces HTML pages. A user can search for particular keywords and obtains a list of results that matches these keywords. The list contains only links to relevant pages with a short description of the content of the page. If the user clicks on a link, the Mobile Google transcoding proxy converts a page matching user's request to the format interpretable by his/her device.

The proxy converts and strips down the page's source into code that better suits devices with limited bandwidth and restricted processing power. It filters the original code, reorganizes it, splits it up, removes some content such as big images and advertisements and inserts its own links into the page. Large pages are paginated into chunks of content. Special links, inserted at the bottom of each chunk, lead to "metapages" of the viewed site (e.g. "links on this page", links that take the user back to the results).

Figure 5.4 presents Sun's Java technology Web site (cf. figure 5.4.1)⁷¹ and its mobile version transcoded by the Mobile Google proxy and displayed on the Nokia 3650 phone. Figure 5.4.2 demonstrates the results of Google search for the keywords “java sun”. If the user clicks on the link displayed as the first search result, he/she gets the start page for Sun's Java. It was automatically split into 5 cards and the navigation links were added at the end of each card (cf. figure 5.4.6). The user can select to display or to hide images. However, if he/she chooses to hide images, the pagination is not provided anymore. The navigation bar was minimized to links starting with a plus sign (cf. figure 5.4.3). If the user clicks on this sign, he gets a complete list of links or hidden elements (cf. figure 5.4.4 and 5.4.5). The keywords for the hidden links are not very descriptive because they are directly taken from

⁷⁰ Cf. <http://mobile.google.com>.

⁷¹ Cf. <http://java.sun.com>.

the Web page. For example the “+skip.. Java .. Solaris” keywords refer to the links at the top of the Web page. “+search.. Home.. APIs” refer to the search tips bar and to links in the navigation bar. Although the links to PDF documents or downloadable software are also shown, PDF documents cannot be displayed and software cannot be downloaded from the page.



1)



Figure 5.4: Results of MobileGoogle transcoding for Sun's Java technology page (http://java.sun.com)

The Mobile Google proxy transcodes HTML pages to WML, XHTML, or cHTML. Instead of accessing a page using the Google search Web site, one can also access it directly by typing the proxy address with some parameters in the following form: "http://wmlproxy.google.com/wmltrans/h=en/g=q/s=0/u=URL@2F/c=0". For obtaining cHTML, "wmltrans" should be replaced by "chtmltrans". "URL" is the URL location of a particular page, "@2F" represents the Unicode character ("%2F" or "/"), "q=" stands for a search term and "c=" is the page number of the page that the proxy breaks up the URL content into. Using the example of java.sun.com as an address that should be transformed, the following link will be created and used by the proxy: "http://wmlproxy.google.com/wmltrans/h=en/g=q/s=0/u=www.java.sun.com @2F/c=0".

From the device-independent perspective, transducing allows for Web access through common URLs. The available functionality of pages, however, may be limited and the sites may lack a good design since even the most intelligent transcoding proxies have restricted possibilities. Intermediaries usually lack specific information about the pages. They cannot guarantee that the transcoded content will be usable on the device that requested it. For example, compressed WAP pages cannot exceed the size of 1.4 KB and the transcoding proxy has to split up the HTML pages so that they fulfill this requirement. This may lead to unusable navigation structures or damaged layouts [FrKuLi01]. Furthermore, some multimedia content or publishing formats may not be supported or downloadable to mobile device.

5.1.5 Transforming

Since pure transcoding approaches only convert Web pages to alternative formats without really adapting the content to the capabilities of devices, transcoding is often mixed with transforming. Transforming modifies the content structure of a site (e.g. by replacing tables with lists or altering navigation elements) and the way of interacting with the content [Schi+02]. Transformation is also a task of a transcoding proxy.

Transcoding proxies use different techniques to render a Web page on a small-screen device. Existing approaches apply various transcoding heuristics [BiGiSu99; BuMoPa00; BuMoPa01; HwSeKi02; LuLa02], semantic annotations [Hori02; HoAbOn03; Naga03; NaShSq01; MeFi03], text summarization techniques [AlRa03; BuMoPa01; BuMoPa00; Buyu+00; RaAl02; RaAlHa01], model recovery [Berg+02; GaBeLa03], or compaction strategies [Cors01].

Transcoding heuristics

The main goal of transcoding heuristics is to preserve the layout of a Web page as much as possible. Seven basic transcoding heuristics with possible enhancements are commonly used: outlining transformation, improved outlining transformation, first sentence elision transformation, restricted first sentence elision transformation, image reduction and elision transformation, table transformation, and indexed segmentation transformation [BiGiSu99; HwSeKi03].

The basic transformations were introduced by Bickmore et al. [BiGiSu99, pp.538-539], whereby their improved versions were proposed by Hwang et al. [HwSeKi02, pp.15-19]. The outlining transformation was developed for paragraphs beginning with headers. It replaces the headers with hyperlinks pointing

to the text extracted from paragraphs. The improved outlining transformation generalizes the outlining transformation mechanism for further combinations of elements. It can be used if pairs consisting of abstract and more detailed elements exist (e.g. `` and ``). The first sentence elision transformation is applied to large blocks of text. It shortens the text block to its first sentence, which is then displayed as a hyperlink. The restricted first sentence elision transformation suppresses the first sentence elision transformation for large blocks of text within a table or for long texts comprising a table. The table transformation is applied instead of the first sentence elision transformation. The table transformation recognizes a table structure and generates one sub-page per cell in the top down, left right order. The image reduction and elision transformation scales down images and produces hyperlinks that refer to the modified images. The indexed segmentation transformation creates a sequence of sub-pages fitting on small displays and connected by hyperlinks from a page.

Hwang et al. [HwSeKi02; HwSeKi03] introduced two additional parameters to enhance the transcoding heuristics: the shrinking factor and information density. The shrinking factor is a ratio between a Web component's size before and after transcoding. The information density expresses the importance of a component and is measured with regard to the amount of content in a component. These parameters are used in the generalized outlining transformation and selective elision transformation. The generalized outlining transformation detects repeated layout patterns and classifies all components into categories depending on the calculated shrinking factors and information density. In the next step, less important components (e.g. components with a high shrinking factor and low information density) are elided. Similarly, the selective elision transformation elides fragments of tables basing on the shrinking factor and information density.

External annotations

Annotation-based transcoding aims at providing machine-understandable description of resources by adding semantics to pages [Hori02; Naga03; NaShSq01]. In this approach, HTML documents are augmented with metadata (annotations) that facilitate the adaptation of Web content by a proxy.

Hori et al. [HoAbOn03; Hori02] proposed external annotations for HTML pages in the form of Resource Description Framework (RDF) files [W3C04d]. The Resource Description Framework (RDF) is a standard, created by the World Wide Web Consortium (W3C), that enables the encoding, exchange and reuse of structured metadata. RDF provides a model for describing resources that are uniquely identifiable by a Uniform Resource Identifier (URI). Resources have certain characteristics that are expressed by Properties and PropertyTypes. PropertyTypes express the relationships of values associated with Resources. Properties are a combination of a Resource, a PropertyType and a value. In RDF, values may be atomic in nature, or may be represented by other Resources, which in turn may have their own Properties. A collection of Properties that refers to the same Resource is called a Description [Klei01; Powe03]. RDF models are usually expressed in XML. An exemplary RDF document presented in listing 5.1 describes the resource "<http://www.w3.org/RDF>":

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.     xmlns:si="http://www.mysite.com/siteinfo#">
4.   <rdf:Description rdf:about="http://www.w3.org/RDF/">
5.     <si:author>World Wide Web Consortium</si:author>
6.     <si:homepage>http://www.w3.org/</si:homepage>
7.   </rdf:Description>
8. </rdf:RDF>
```

Listing 5.1: Exemplary RDF document

The document expresses 2 statements: “The author of the page <http://www.w3.org/RDF> is the World Wide Web Consortium” and “The homepage of <http://www.w3.org/RDF> is <http://www.w3.org/>”. Both statements are a combination of a subject (Resource), a predicate (Property) and an object (Property value). For example, in the first statement the subject is <http://www.w3.org/RDF>, the predicate is the author, and the value is W3C.

Annotations are usually stored in a repository and are not embedded in pages in order to separate content from presentation. XPath [W3C04j] and XPointer [W3C02f] are used to associate portions of a document with appropriate annotations. XPath is a set of syntax rules for addressing parts of an XML document. XPointer is an extension of XPath, suited for linking and addressing internal parts of XML documents. For example, the expression `/BODY/IMG[2]` refers to the second image placed in the BODY element. Specific vocabulary with three types of annotations (alternatives, splitting hints, and selection criteria) was specified [Hori+00, pp. 200-201]. Alternatives provide a list of substitute representations for a particular element. For instance, a JPEG image may have its alternatives in the WBMP or GIF format. Splitting hints specify the groups of elements that should be considered as one logical unit and therefore should not be split on smaller displays. They also determine the break points. Selection criteria contain information that helps the transcoding proxy to choose the most appropriate content for a particular device. They include the role and importance of annotated elements or expected device capabilities. Basing on these annotations, the elements with low importance may be omitted. The proxy is also able to render navigational components and layouts properly. For example, tables often serve as positioning elements. If a table is annotated in a role of “layout”, it will be ignored on small, wireless devices. For creating annotations, a specific authoring tool was developed [HoAbOn03].

Model recovery

Model recovery approaches [Berg+02; GaBeLa03] aim at inferring abstract models from existing applications, for example from Web pages. Simple identification of UI elements in HTML pages is considered as being inadequate: model recovery also takes into account semantic relationships between elements. Therefore, instead of parsing HTML documents and building trees of UI elements, model recovery techniques use domain-specific knowledge encoded in the form of sets of rules.

Deduction rules facilitate the aggregation of UI elements by assigning semantic to them. Mutual exclusion rules specify types of conflicting deductions. Scoring rules establish preferences based on the features of interactors⁷². The process of rules discovery is based on observations of HTML pages. A deduction rule states, for example, that captions and hints always occur in the neighborhood of their corresponding interactors. Scoring rules typically take into account the relative distance, orientation, and positions of interactors [GaBeLa03, p. 73]. A model recovery technique was applied in Platform-Independent Model for Applications (PIMA) [Bana+00] and in Multi-Device Authoring Technology (MDAT) [Bana+04].

Summarization techniques

Summarization techniques, usually relevant in the context of information retrieval, are applied to pre-processed Web pages. The pre-processing stages may differ in various approaches. In WEST, pages displayed in a browser [Björ+99] are first chunked into cards collected in decks. In Power Browser [BuPaMo01, p. 652], a page is partitioned into Semantic Textual Units (STUs). Exemplary STUs elements are lists, tables or paragraphs. STUs may be nested. The extraction of STUs is based on a structural analysis of Web pages and is called a “macro-level summarization” as opposed to a “micro-level summarization” which uses information retrieval techniques for summarization [BuPaMo01].

In the page pre-processing stage, an HTML site is usually converted to a DOM tree representation and corrected for syntax errors [cf. BoKaMi04; ChMaZh03; Gupt+03; Kaas+00; MiKaBo04]. The tree is then used for the identification of content blocks (i.e. groups of objects representing the semantic structure of the page - headers, footers, navigation side bars, etc.) [ChMaTh03] or for the removal of unnecessary elements (advertisements, scripts, styles) [Gupt+03].

In automatic summarization, the content is extracted from an information source. Its most important parts are presented to the user in a condensed form, in a manner sensitive to the user’s or application’s needs. Common text summarization methods use natural language processing (NLP) based on the computational linguistics, and include the following processes for mobile devices [Alam03; ZhHeMi04]:

- keywords generation - certain words are extracted from a text, based on the evaluation of the importance of each word. This importance is measured with the help of Term Frequency-Inverse Document Frequency (TF/IDF) measure that assumes that a word is essential if it occurs frequently within a text, but infrequently in a larger collection of documents [Seki02]. The TF/IDF value is computed as:

$$TF / IDF = \text{frequency of a term in a document} * \log\left(\frac{\text{number of all documents}}{\text{number of documents with a term}}\right)$$

⁷² Interactors are UI elements such as checkboxes, text fields, buttons, etc.

The collection of documents, which in the case of a Web page is the World Wide Web, can only be approximated in the form of a dictionary containing the words and its importance [BuMoPa01, pp. 655-656].

- summary generation - the most important sentences are extracted from a text and displayed. Sentences with the greatest number of frequently occurring distinct words, found in the greatest physical proximity to each other, are considered as crucial [cf. Luhn58].
- keyword/summary method - combines both described methods. Buyukkokten et al. found that this approach guarantees the best performance and user experience while searching with the help of Personal Digital Assistants [BuMoPa01]. However, it should be noted that for some pages, the summarization methods may provide unusable results [Alam+03].

Compaction methods

Compaction techniques neither elide text fragments, nor reduce the amount of information by applying summarization approaches. They aim at keeping the loss of content to an absolute minimum. Corston [Cors01], for example, described a compaction approach for displaying emails on mobile devices based on three levels of compaction: LONGFORM, COMPRESSEDFORM, and CASENORMALIZEDFORM. The LONGFORM is obtained by taking a string that does not contain a high portion of punctuation characters, and reducing multiple leading spaces to a single leading space. The COMPRESSEDFORM is a compacted representation of a string, and the CASENORMALIZEDFORM represents a normalized version of the previous form with spaces removed and the first letters capitalized. The COMPRESSEDFORM is produced with the help of linguistic text compaction strategies by using the character removal technique based on experiments with native speakers. The output can be generated for four languages (English, German, French, and Spanish) and may look like this:

Compacted version:

PrblmOfAutmtcSmmrztnPssVrtOfTghChllngsInBthNLUndrstdng&Gnrtn.

Source text:

The problem of automatic summarization possesses a variety of tough challenges in both NL understanding and generation.

Although the initial user experiences were evaluated as positive, this approach requires really good deciphering ability and may result in relatively long reading times or incorrect interpretations.

5.1.6 Comparison of methods for content adaptation

Content adaptation methods can be evaluated using different sets of criteria. The Device Independence Principles can be helpful in accessing the functionality of methods and their grade of device independence, but they do not include factors such as the complexity of a method or possible

trade-offs between the development cost and the quality of approaches. Additional aspects are therefore assessed using device independence authoring challenges.

The evaluation and comparison of the approaches is provided in tables 5.1 and 5.2. Table 5.1 presents a generic comparison of adaptation methods with regard to the Device Independence Principles. It should be noted that the comparison is not of a quantitative nature and should solely offer a guide for content developers or managers interested in using one of these methods. Content developers should be aware of the fact that different criteria can have different importance weights, and that the chosen method can be highly dependent on the type of application. For example, if a mobile tourist guide application has to be programmed, scaling might be a good choice because users will probably have to view large maps or images, but only a limited number of textual information. If the application is simple and well-structured from its very nature and should be used by as many users as possible, transcoding or transforming will take good care of the adaptation. Similarly, if the quality of pages is important and the number of clients is limited to PDAs users, even manual authoring may be a feasible solution.

Approach Criteria	Manual authoring	Scaling	Transducing	Transforming
DIP 1: Device independent access	**	**	**	**
DIP 2: Device independent Web page identifiers	*	**/**	***	***
DIP 3: Functionality	***	**	**	***
DIP 4: Incompatible access mechanism	*	*	**	**
DIP 5: Harmonization	***	**	*/**	**/**
Evaluation scores: *: no support offered **: functionality only partially supported ***: functionality supported				

Table 5.1: Evaluation of content adaptation methods with regard to Device Independence Principles (DIPs) [own research]

Table 5.2 evaluates the authoring techniques against a set of Device Independence Authoring Challenges (DIACs). These challenges focus on the authoring perspective and review primarily the quality, feasibility, and usability of the approaches. The Device Independence Principles, on the contrary, address factors that are relevant from the user's perspective. The assessment of adaptation techniques in terms of authoring challenges shows certain trade-offs between the complexity and comprehensibility of these methods and their affordability. Affordability and complexity are correlated: the more comprehensive the approach, the higher development and maintenance costs it generates.

Technique	Manual authoring	Scaling	Transducing	Transforming
Challenge/ Evaluation criteria				
Provide comprehensive scope				
DIAC-3.1: Application scope	**	**	***	***
DIAC-3.19; 3.20: Integration of device-dependent and independent content	***	*	***	***
DIAC-3.28: Range of complexity	***	***	***	****
Support smooth extensibility				
DIAC-3.2: Extensible capabilities	****	-	****	****
DIAC-3.29: Scalability of complexity	****	***	****	****
Support simplicity				
DIAC-3.4: Simplicity	***	***	**	**
DIAC-3.11: Simple content	***	**	**	**
Support delivery context variability				
DIAC-3.5: Navigation variability	*	*	**	***
DIAC-3.6: Organization variability	*	*	**	***
DIAC-3.7: Media variability	**	*	***	***
Support author specified variability				
DIAC-3.12: Text content variety	**	*	*	***
DIAC-3.13: Media resource variety	**	*	**	***
DIAC-3.14; 3.15: Media resource specification & selection	*	*	*	***
DIAC-3.30; 3.31: Aggregation & decomposition	*	*	**	***
DIAC-4.3: Layout variety	**	**	**	***
Affordability				
DIAC-3.3: Affordability (cost)	**	***	***	***
DIAC-3.33: Reusing existing applications	*	***	***	****
DIAC-6.5: Minimization of effort	*	***	***	***
DIAC-6.13: Separation of device-dependent and device-independent material	*	*	**	***
DIAC-6.6: Abstraction of device knowledge	*	**	***	****
DIAC-6.10: Scalability of effort/quality	*	**	***	***
Scale: * : non existing; ** : low; *** : middle; **** : high; - : does not apply to this method				

Table 5.2: Comparison of content adaptation methods with the help of Device Independence Authoring Challenges (DIACs) [own research]

5.2 Categories of adaptation approaches

Since the mid-nineties, the number of frameworks for tailoring data to various devices has rapidly increased, proving the growing interest in device-independent content delivery. The proposed approaches belong to three categories, depending on the controller of the adaptation process: intermediate, client-side, and server-side adaptation. These mechanisms are also called adaptation processors, and are defined as frameworks that get one form of content as input and produce an alternative form as output [W3C04].

5.2.1 Intermediate adaptation

In intermediate adaptation (cf. figure 5.5) an intermediary (a proxy) is placed between an application server and a client, and it controls the adaptation process. Proxies can be remote or can be applied as a combined local/remote configuration [StPa02]. The local/remote proxy approach takes care of bandwidth problems, while remote proxy approach is mainly concerned with reformatting Web pages to better match the display capabilities of a device.

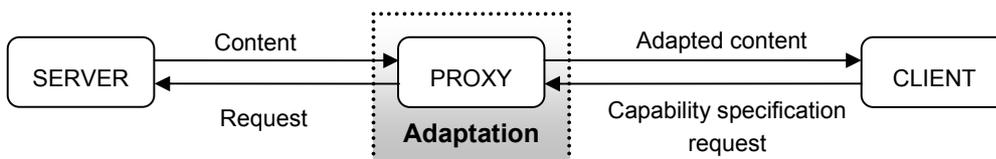


Figure 5.5: Intermediate adaptation [Butl+01, p.4]

The proxy converts markups generated by a server to new formats, depending on the characteristics of a client. Both, the client and the server, may not be aware of the proxy presence and support for new formats can be introduced by adding features to the proxy. The proxy should theoretically enable a broad and on-the-fly access to the Web. In practice, proxies may encounter severe problems while transforming complex or badly designed pages [Butl+02; FrKuLi01]. The adaptation quality is highly dependent on the knowledge about device capabilities, implemented algorithms, and the presence of additional metadata, providing adaptation hints.

5.2.2 Client-side adaptation

Client-side adaptation (cf. figure 5.6) is not a commonly used method, because the adaptation code needs an access to the capabilities of devices, and the devices have to be quite powerful. The adaptation is performed entirely on the client that decides about the final appearance of the content. The content is sent to a client in a native format (e.g. HTML). The client can then reformat it or omit some parts of the content using special style sheets. The adaptation is usually applied to images (resizing), fonts (substitution with other fonts or changing the font size), and forms (reducing the visualization of form controls, repositioning form controls, transforming form controls, splitting the form) [W3C04].

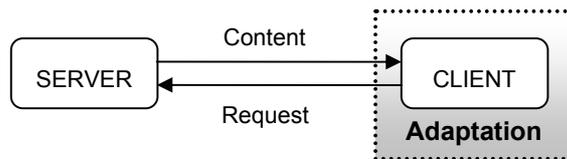


Figure 5.6: Client-based adaptation [Butl+01, p.4]

In this approach, no bandwidth is saved, since the whole page is transmitted to the client together with appropriate data for presentation logic. The reformatting is computationally intensive and has to be performed entirely on the client. This method can be therefore successfully applied by HTML browsers or modern devices equipped with powerful processors.

5.2.3 Server-side adaptation

Server-side adaptation (cf. figure 5.7), in which a server is in charge of content delivery, gained the widest acceptance of authors [Butl+02]. Two adaptation mechanisms on the server-side are possible. The appropriate markup language can be dynamically generated on the server, depending on the characteristics obtained from the device. Alternatively, a version that optimally suits the delivery context may be selected from the existing pages and delivered to the device [W3C04].

The server can respond to a user's request, and can deliver radically different content. The differences are not only related to the markup, but also to the amount of information, presentation style, navigation and layout. The ability to obtain precise characteristics of devices determines the quality of the server-side adaptation.

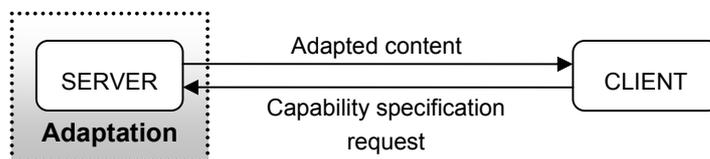


Figure 5.7: Server-based adaptation [Butl+01, p.4]

6

Related work in the area of content adaptation to mobile devices

The shift from Web-based information services towards pervasive services that are accessible from various devices contributed to the emergence of multiple frameworks. These frameworks aimed at the generation of user interfaces that offer a similar functionality and presentation for heterogeneous devices. Device-independent representations of content and adaptation approaches proved to be an effective solution for the problems of varying device capabilities and multiplicity of supported content formats.

This chapter outlines some selected approaches for device-independent content delivery. Its structure is based on the categorization of frameworks according to the controller of the adaptation process. Section 6.1 presents Cascading Style Sheets and Small Screen Rendering; a CSS-based Opera's technology for rendering existing HTML pages on mobile devices. Section 6.2 focuses on intermediate adaptation and describes the architectures of well-known frameworks such as m-Links, Web Intermediaries, Power Browser, or WebAlchemist. The next section introduces approaches for server-side adaptation and divides this adaptation category into frameworks based on XML/XSLT technology, User Interface Description Languages (UIDL), and architectures created around the Model-View-Controller (MVC) design pattern. The last section provides a comparison of the most important approaches belonging to different adaptation categories.

6.1 Client-side adaptation approaches

Client-side adaptation is mainly used in desktop-based applications because this type of adaptation requires a lot of computational power. For mobile devices equipped with robust processors two technologies are feasible: Cascading Style Sheets (CSS) and XML/XSLT transformation. The second approach is usually encountered in server-side adaptation and will therefore be described in section 6.3.1.

6.1.1 CSS and related technologies

Cascading Style Sheets (CSS), currently in version 3 (CSS3), are the most popular example of client-side adaptation [W3C06]. Authors can use style sheets to define different presentation rules for various media types. In this way, presentation is entirely separated from content and various browsers may deliver different versions of the same page, depending on the device type. If the presentation needs to be changed, transformations are made in the style sheet and modifications are immediately applied to all pages. CSS allows styling of languages based on XML such as HTML, XHTML, or Scalable Vector

Graphics [W3C03c]. Style sheets take care of fonts, text attributes, borders, margin alignment, colors, positioning of elements, etc.

For small devices, the W3C Consortium defined CSS Mobile Profile (CSS MP) - a subset of CSS, omitting features that are not applicable to mobile devices [W3C02a]. For example, CSS MP does not support such CSS properties as cue, outline, page, pause, position, etc. [cf. W3C02a]

CSS can be used to change the presentation of content on different devices. However, browsers processing CSS need information about the delivery context, or alternatively, only one style sheet per page and device has to be specified. CSS offers two mechanisms that help to select and apply the appropriate style sheet from a CSS collection depending on the delivery context. The “Media Types” method is already supported by different manufacturers of mobile devices, support for Media Queries is still in progress.

Media Types

Since the second version of CSS (CSS2), the Media Types⁷³ mechanism is used for more accurate rendering of content on certain types of devices. It is, for example, possible to elide fragments of pages such as navigation columns and advertisements or to scale images. The elision mechanism is particularly interesting – by using the CSS property “display” with the “none” value certain page elements will not be rendered. Although this property allows slight manipulations of the document’s structure, it wastes bandwidth. Once the entire content is delivered to the device, some parts of it can be eliminated with the help of style sheets. Table 6.1 lists the available media types.

Media Type	Usage
All	All devices
Braille	Braille tactile feedback devices
Embossed	Paged Braille printers
Handheld	Handheld devices (small screen, limited bandwidth)
Print	Paged documents or documents viewed in print preview mode
Projection	Projected presentations
Screen	Color computer screens
Speech	Speech synthesizers
Tty	Media using a fixed-pitch character grid (teletypes, terminals, portable devices with small displays)
Tv	Television-type devices (low resolution, color, limited scrolling ability, sound)

Table 6.1: Media types for CSS

⁷³ Cf. <http://www.w3.org/TR/REC-CSS2/media.html>.

In a style sheet, media types, to which particular properties apply, are specified by the @media rule:

```
1. @media screen
2. {
3.   BODY { font-size: 12pt }
4.   H3 { font-size: 8pt; color: silver }
5. }
6. @media handheld
7. {
8.   BODY { font-size: 8px }
9.   H3 {display: none }
10. }
```

Listing 6.1: Example of media types

In the example shown in listing 6.1, the font-size for color computer screens is set to 12 points and for handheld devices to 8 pixels. The text enclosed by H3 tags will not be displayed on mobile devices. The media type “handheld” refers to all wireless devices and will result in the same presentation on all mobile devices. However, a style sheet defining the general appearance of the content for wireless devices may be extended by adding more specific style sheets for particular types of devices.

Media Queries

CSS3 Media Queries [W3C02d] allow Web designers to specify presentation options based on the distinctiveness of target devices. Media Queries represent an extension of the Media Types mechanism. A Media Query contains information about a media type and media features. Media types applied in a Media Query are the same as in the Media Types approach, media features refer to relevant characteristics of devices: their widths, heights, supported colors, resolutions, and device-aspect-ratios.

```
1. @media handheld and (min-resolution:300dpi) and (device-aspect-ratio:
2. 1289/760) and (color)
3. {
4.   BODY { font-size: 8px; background-color:#A3B0F3; }
5. }
```

Listing 6.2: Media queries

In the example introduced in listing 6.2, the properties for the tag <BODY> apply to a color handheld device with a minimal resolution of 300 dots per inch (dpi) and device-aspect ratio of 1289/760.

For the process of mobile content adaptation, the CSS technology has severe disadvantages. It can only be applied to XHTML or HTML. Mobile browsers, for example, are not able to render XML with the help of CSS. Furthermore, it is impossible to influence the structure of a document by changing the order of appearance of certain elements or by creating new elements from the existing ones. However, most wireless devices are currently equipped with WAP browsers supporting XHTML and CSS. Therefore, both technologies will probably dominate the mobile browsing within the next years.

Opera's Small-Screen Rendering (SSR)

Since 2002, Opera Software⁷⁴ uses its proprietary technology called Small-Screen Rendering (SSR) for client-side adaptation on devices with limited display capabilities. Small-Screen Rendering is based on CSS and Media Types. It transforms Web pages and the user can view them on small screens without the need for horizontal scrolling. Instead of this, he/she has to scroll down to see more content and can use the zooming functionality to scale the page.

The figure illustrates the Small-Screen Rendering (SSR) technology by comparing a desktop view of the Nokia.co.uk website with its small-screen rendering. The desktop view (right) shows a standard layout with a navigation bar, a main banner for the Nokia 3250, and several columns of content. The small-screen rendering (left) shows the same content adapted for a smaller screen, with a vertical layout and large, bold numbers 1, 2, 3, and 4 highlighting specific sections. The small-screen rendering uses a vertical layout, with content stacked on top of each other. The desktop version uses a horizontal layout, with content arranged side-by-side. The small-screen rendering also features a search bar and a navigation menu at the top, and a footer with links to various sections of the website.

Figure 6.1: Small-Screen Rendering for Nokia.co.uk page

⁷⁴ <http://www.opera.com>.

The content and functionality of Web pages do not change, only the layout is modified. Opera browsers are currently available for Nokia 7650, Nokia 3650, Nokia N-Gage, Siemens SX1, Sony Ericsson P800, and P900⁷⁵.

The adaptation process is computationally intensive. It is only possible for devices with the Opera browser installed on them. Furthermore, HTML documents have to follow certain rules (e.g. a separation of content and presentation is required, the pages cannot possess any frames, etc.) to be rendered properly. Figure 6.1 presents the entry page of Nokia.com transformed by SSR. Exactly the same content is displayed on a mobile device (left) and desktop computer (right), the images were scaled down or omitted, the text size was reduced, and empty spaces were eliminated.

Although Opera calls Small-Screen Rendering “an absolutely phenomenal technology” [Fest02], it is actually a relatively simple CSS style sheet with appropriate media rules specified. In Appendix 3, a style sheet developed by the author achieves results similar to SSR⁷⁶. Although SSR and CSS are generally not perfectly suited for mobile devices (excessive horizontal scrolling, possible problems with overflows, etc.), they certainly prove that device-independent content delivery may be achieved with a very simple technology.

6.2 Intermediate adaptation approaches

Intermediate adaptation is applied in the transformation of existing Web documents into pages viewable on different clients. Neither the server, providing the original content, nor the client requesting it has to be aware of the intermediary’s existence. However, not all pages can be transformed properly because transforming proxies often lack specific knowledge about converted sites. This section outlines some exemplary work in the area of intermediate adaptation and presents different concepts applied to enhance the adaptation quality.

6.2.1 Top Gun Wingman

Top Gun Wingman [Fox+98; Fox+98a] was the first graphical Web browser for the 3Com PalmPilot. The browser relies on the programming model for proxy-based applications called TACC [Fox97]. TACC can be decrypted as transformation (distillation, filtering, format conversion, etc.), aggregation (collecting data from various sources), caching, and customization (management of user profiles). For each particular task (e.g. scaling of images, conversion of formats) the so-called worker is responsible. Workers can be composed into a chain, forming a new application. The main components of Top Gun Wingman proxy are four TACC workers: HTML processor, image processor, zip processor, and aggregator request service. By moving all the complexity of Web browsing to the TACC proxy, the functionality of the mobile browser was reduced to the rendering of primitive data types (images, data types, and links).

⁷⁵ Cf. <http://www.opera.com/products/mobile/smallscreen/>.

⁷⁶ The style sheet was tested with Netscape and Mozilla browsers.

The image processor converts GIF and JPEG images to intermediate bitmap forms, scales them, color-quantizes, dithers, and finally provides images in PalmPilot's format (Tbmp) or enhanced 2-bit-per-pixel format. The HTML processor parses HTML and produces a client-dependent, intermediate-form page layout. Images encountered during parsing are fetched and dispatched to the image processor. The parsed pages are subsequently converted to a tokenized markup that allows sending the whole content as a single object. The zip processor formats the listing of a zip archive as HTML. The HTML code is also parsed by the HTML processor. It is converted to the developed, tokenized markup. The aggregator request service produces content-aggregation applications. An aggregator is defined as a PalmPilot user interface that allows the user to specify his/her requests by writing some text. The aggregator queries Web sites for the required content, collects it, and formats the results for the presentation on wireless devices.

Wingman browser resembles desktop browsers and possesses features such as bookmarks, a local, user-controllable cache and history cache. The user can furthermore benefit from the zooming functionality that allows increasing images to their original sizes.

6.2.2 WebAlchemist

WebAlchemist [HwSeKi02; HwSeKi03] is a HTTP proxy that applies advanced transcoding heuristics (described in section 5.1.4) and consists of four modules: a HTML Tokenizer, a Grammar Corrector, an Internal Representation Generator, and a Transcoding Manager. The HTML Tokenizer module recognizes HTML tags and converts them into tokenized strings. The Grammar Corrector module corrects syntactic errors in HTML pages. The Internal Representation Generator converts the corrected HTML syntax, received as tokenized strings, into its tree-based representation. The Transcoding Manager performs the transcoding process with the help of transcoding heuristics. It also computes the shrinking factor and the information density for each page and device.

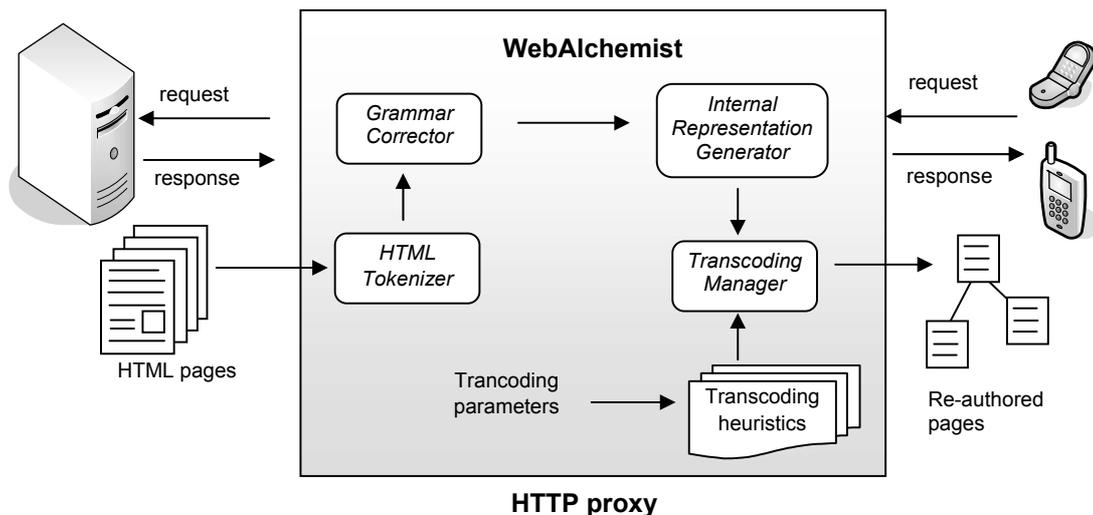


Figure 6.2: WebAlchemist transcoding system [HwSeKi02]

The following default transcoding sequence proved to guarantee high-quality transcoding and is used in the WebAlchemist system:

- the improved outlining transformation
- the generalized outlining transformation
- the selective elision transformation
- the restricted first sentence elision transformation
- the image reduction and elision transformation
- the indexed segmentation transformation

After transcoding, pages are converted back from their tree representations to the original source formats. Figure 6.2 outlines the architecture of the WebAlchemist system.

WebAlchemist was designed only for handheld devices with browsers supporting HTML. It does not handle the conversion of HTML into other markup languages. Instead of this, it tries to produce pages that fit on small screens by reducing the size of images and converting the original structure of HTML pages to compositions that make it possible to avoid horizontal scrolling.

6.2.3 Web Intermediaries (WBI)

The annotation-based transcoding framework applied by Hori [Hori+00] uses a programmable proxy server named Web Intermediaries (WBI)⁷⁷. WBI receives HTTP requests from clients and generates HTTP responses using a set of specific plug-ins. All plug-ins consist of three components: Monitors, Editors, and Generators. Monitors are responsible for transactions monitoring, Editors modify incoming documents or outgoing responses, and Generators generate appropriate responses. The process of annotation-based transcoding with a page-splitting plug-in is depicted in figure 6.3. Upon a request, a page is retrieved from a content server. The Editor component looks for appropriate annotation files and retrieves them. If no annotation files exist, the original page is sent to the client. The Generator component takes into account the device capabilities retrieved from HTTP headers, processes the page according to the annotations (in the case of the page splitting plug-in, it divides the page into smaller sub-pages), and returns the output to the client.

The described framework became part of a commercial IBM software - the WebSphere Transcoding Publisher⁷⁸. This tool enables the transformation of HTML documents into WML, cHTML, HDML, VoiceXML, and ClipperML for Palm.Net devices as well as image transcoding [IBM04].

⁷⁷ <http://www.almaden.ibm.com/cs/wbi>.

⁷⁸ http://www-306.ibm.com/software/pervasive/transcoding_publisher.

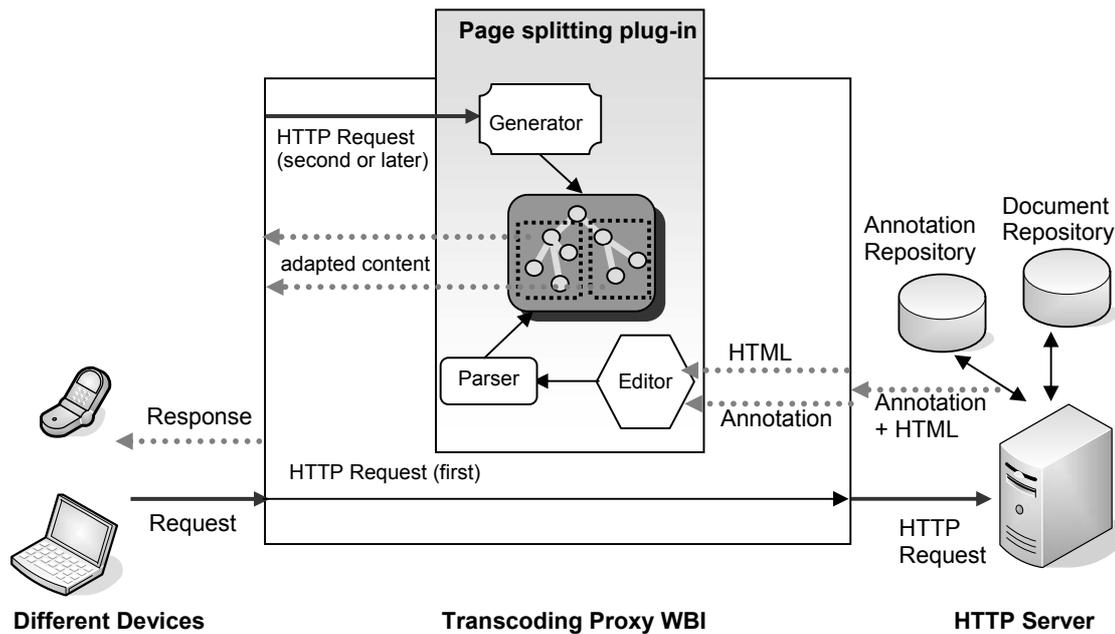


Figure 6.3: Web Intermediaries [Hori+00, p. 203]

6.2.4 Power Browser

Power Browser [BuMoPa00; BuMoPa01; BuPaMo01; Buyu+00] is a proxy-based Web browser designated for PDAs⁷⁹. It uses text summarization methods to present information on small devices. The Web page is broken into Semantic Textual Units (STUs) that can be hidden, partially displayed, fully visible, or summarized. The units can be shown in five progressive modes: incremental, all, keywords, summary, and keyword/summary. In the incremental method, the unit is disclosed gradually: at the beginning the first line is displayed, then the first three lines are visible, and at the end the whole unit is presented. The "all" disclosure method allows showing the whole unit at once. The keywords method displays keywords at the beginning, then the first three lines are shown and subsequently the whole unit is presented. In the summary method, first the summary is shown and then the whole unit is displayed. The keyword/summary method consists of three states: in the first state keywords are presented, in the second state the summary is displayed, and in the third state the entire unit is shown. The authors came to the conclusion that the keyword/summary method is the most effective one.

⁷⁹ Power Browser can be found under: <http://www-diglib.stanford.edu/~testbed/doc2/PowerBrowsing/download.html>.

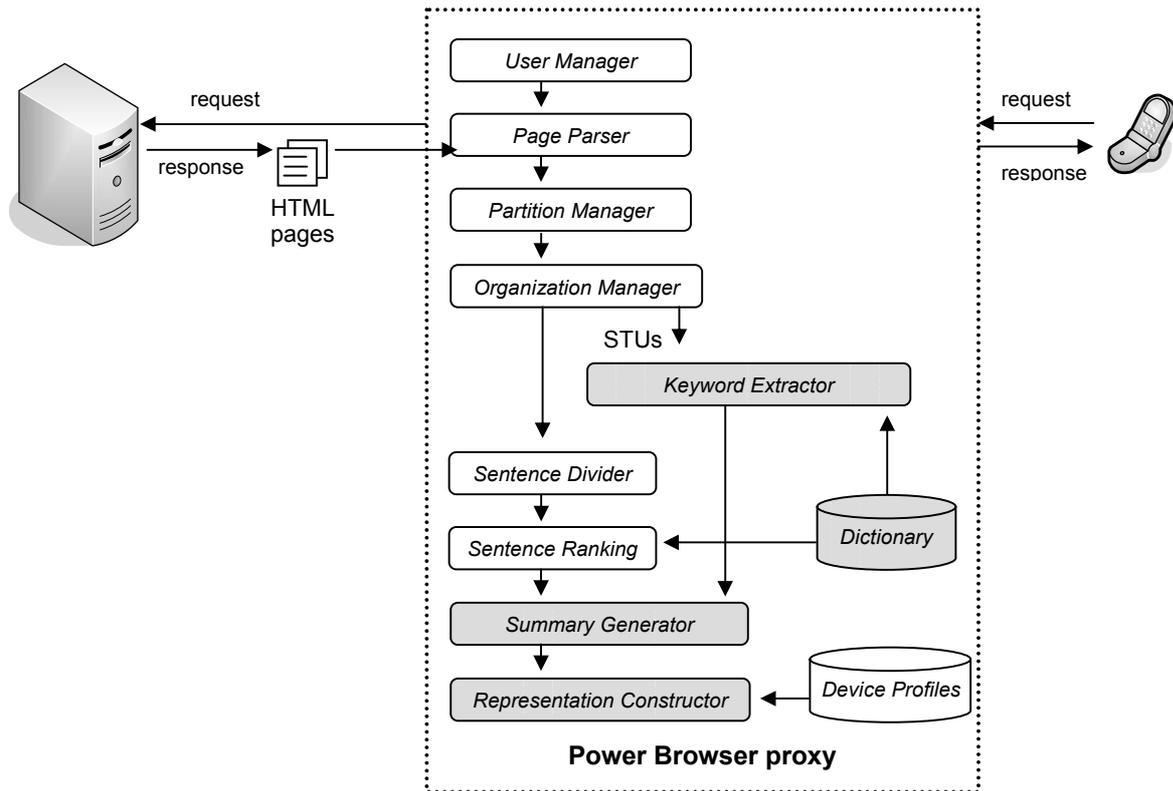


Figure 6.4: Power Browser proxy [BuMoPa01, p. 655]

Figure 6.4 presents the architecture of the Power Browser proxy. The proxy consists of a set of managers that are responsible for various tasks. User Manager is in charge of the management of user preferences (e.g. preferred disclosure method) and keeps track of the information that was requested by the user's PDA. Page Parser extracts all tokens from the downloaded site, Partition Managers recognizes the textual units. Organization Manager creates a hierarchy of STUs. Sentence Divider, Sentence Ranking, Keyword Extractor, and Summary Generator are responsible for the generation of appropriate views. The Keyword Extractor scans all words in each STU and chooses the most important from them by comparing the selected keywords with the words and their respective importance weights stored in the Dictionary database. Representation Constructor communicates with the database that contains device profiles and composes appropriate displays depending on the device features. The generated view is then sent to the device that requested it.

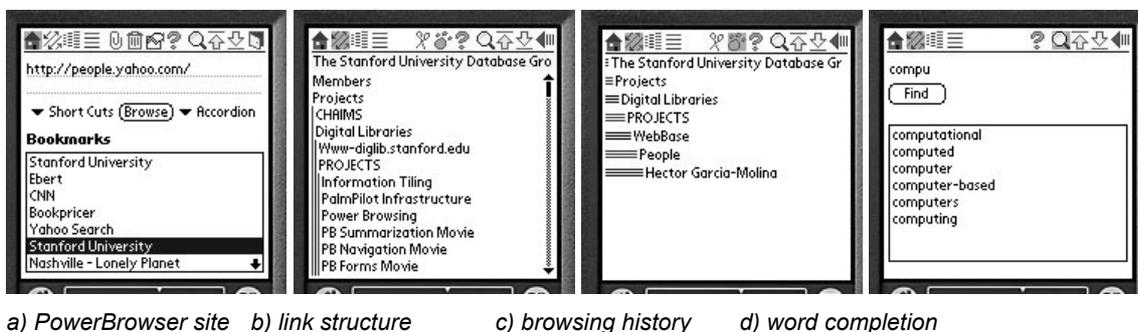


Figure 6.5: Power Browser navigation methods [BuMoPa01]

Power Browser provides a set of techniques for interacting with the Web that facilitates navigation, browsing, and searching [BuMoPa00; Buyu+00]. Figure 6.5 displays some of these methods applied for navigation and searching. The user can enter the URL of a page, define his/her own bookmarks, or find an URL by using a search engine (cf. figure 6.5a). In order to ease navigation, Power Browser extracts informative link information from pages and presents them on one page (cf. figure 6.5b). The links are displayed in a tree form and each node can be expanded if the user needs more details. The user can move to the full text of a link by clicking on it. He/she can also see the browsing history (cf. figure 6.5c). Moreover, Power Browser provides automated local site search functionality for all pages. The user can also enter the first letters of a keyword and is able to see a list of words matching these letters (see figure 6.5d). Additionally, she/he can see pages that match the chosen keyword.

6.2.5 Web Digestor and m-Links

Digestor [BiSc97] is a proxy-based re-authoring system that parses a Web page into its tree representation (the so-called Abstract Syntax Tree - AST) and applies a set of transformations to this tree, using transformation heuristics [BiGiSu99]. The transformed tree is converted into a device-specific format: HTML or HDML. This approach is similar to the methods used in WebAlchemist, but the applied transcoding heuristics differ.

Based on the experiences with Digestor, the Mobile Links (m-Links) system was designed [Schi+01; Schi+02, Trev+01]. In Digestor, the transduction of Web sites often generates complicated navigation structures that are difficult to understand. m-Links changes the browsing experience by providing specific navigation structures. Additionally, m-Links adds actions on the link content (e.g. reading, opening, sending, or printing). Presenting Web pages in the form of links leads to problems with link naming, unlinked data, link overload, and unlimited content types. Extracted links should have meaningful labels that will be helpful in the navigation process and will facilitate searching. In m-Links, the quality of links' names is improved by the application of special link-naming algorithms. Useful data that are not explicitly linked, such as phone numbers or addresses, are extracted with the help of data detectors. To cope with link overload on one page, links are grouped into categories. Furthermore, different actions can be performed on the linked content depending on link's attributes such as its MIME types.

Figure 6.6 shows an example of the m-Links transcoding result for FX Palo Alto Laboratory's site that displays the major publications of the institute. The links are preceded by folder icons (cf. figure 6.6a). Document icons identify links to non-HTML content (PDF, multimedia). By selecting a link and pressing the "Tools" softkey, the user obtains a list of all actions available for this link (figure 6.6b). For example, the "About" action returns a page with the properties of the link's target such as its URL, size, and MIME type. The selected links and actions are identified by an arrow in the front of the chosen item.

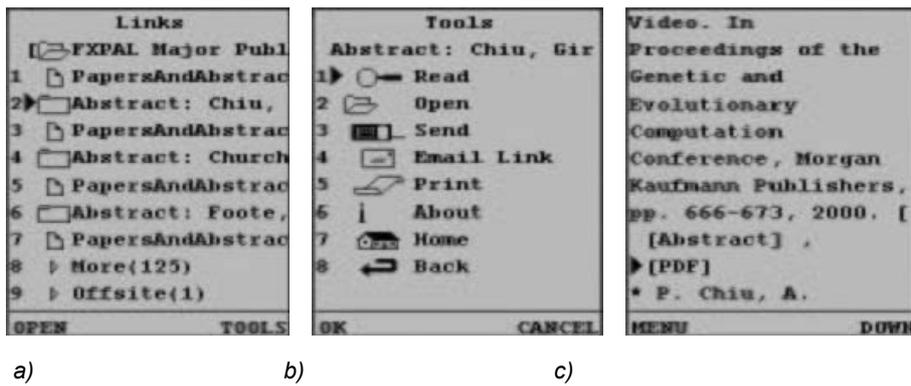


Figure 6.6: m-Links interface on NeoPoint 1000 [Schi+02, p.40]

The m-Links system consists of three main components: Link Engine, Service Manager, and User Interface Generator (cf. figure 6.7). The Link Engine, comprising three elements, is responsible for the processing of Web pages into a collection-of-links data structure that is stored in a cache. In the first step, the HTML parser element creates a tree of HTML tags. The tree is then passed to data detectors that identify addresses and phone numbers and insert additional links providing this information. Subsequently, the Link Engine extracts links and gives them appropriate labels using a link-naming algorithm. The authors of this algorithm assume that the URL of a link represents the lowest quality label and the title of a document, to which the link leads, is the highest quality label. Alternatively, the link's anchor text, the alt-text associated with the image, or the link's relative path may be used as labels. The Link Engine also categorizes links. The Service Manager generates the interface displaying possible actions for a link. The action list is constructed using device properties and link's attributes. The User Interface Generator produces appropriate mobile markups. It is able to generate HDML, cHTML, WML, and HTML. The User Interface Generator recognizes the device requesting a page, determines a suitable markup language, and dispatches the request to a markup handler. The handler applies screen templates to produce appropriate markups.

Although the m-Link system enables faster browsing, users may sometimes get lost in the thicket of links. This issue was addressed in the Tools menu by introducing the read-around feature. This feature displays text surrounding the selected link. Furthermore, transcoded pages consisting of collections of links (e.g. portals) may lead to mobile pages with hundreds of links, decreasing the navigation speed. In such cases, systems like Digester would be more appropriate than the m-Links transcoding proxy.

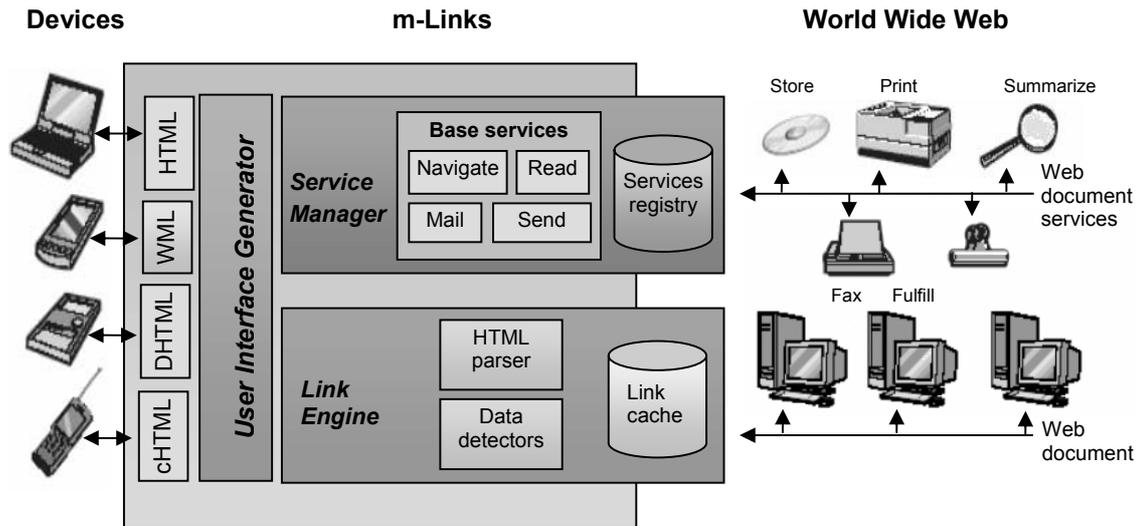


Figure 6.7: *m-Links architecture [Schi+02, p.41]*

6.2.6 Web clipping

Web clipping is a proprietary technology supported only by Palm handhelds. It was initially used by Palm on its Palm VII handheld devices [Hill01]. Web clipping aims at the optimization of data transfer and the best possible presentation of Web pages on devices with limited displaying capabilities. The idea behind Web clipping is to preserve handheld resources and to save the bandwidth by extracting any static data from Web pages such as graphics, photos, or unnecessary text. Special applications (the so-called Palm Query Application, PQA) and all static content has to be downloaded and stored on the device.

A Palm Query Application is created by compiling HTML pages with Web Clipping Application Builder (WCA Builder) to the Compressed Markup Language (CML) format. Links within a PQA document can either refer to other pages within the application or to the documents or scripts residing on a publicly available Web server. The generated PQA file has to be installed on a handheld as the Palm records database. This database type can be opened with Web Clipping Application Viewer. A PDA with the PQA file on it makes a wireless connection to a proxy server at the Palm.Net data center in order to retrieve the dynamic content. The proxy server translates HTML pages into the Compressed Markup Language format and sends them back to the device. CML is then rendered by the WCA Viewer on a Palm screen. Web clipping is therefore an example of intermediate adaptation, where a proxy is in charge of the transformation. The process is illustrated in figure 6.8.

Web clipping applications are written in a subset of HTML 3.2 and can be authored very quickly by HTML designers. They do not, however, support features that could influence the performance of applications such as, for example, style sheets, JavaScript, nested tables, frames, or cookies. It is possible to retrieve maximally 500 KB of compressed data, thus PQA applications have to be designed carefully. The Palm VII comes with a number of pre-installed, useful PQA's including those from Amazon.com, Yahoo!, or MapQuest.

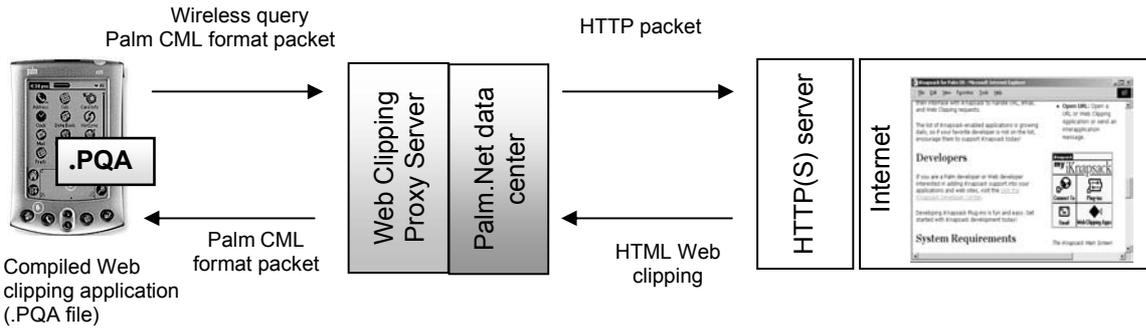


Figure 6.8: Web clipping architecture [own illustration]

6.2.7 WebViews

WebViews [FrKuLi01] is a very interesting and unconventional approach for delivering Web content to wireless devices. A user can record his/her browsing through a page on a desktop computer using the WebViews Recorder applet. He/she may select page fragments that are interesting for him/her. Such fragments are named Web Views and their specifications are saved in a database. All navigation actions (links traversed, user inputs, etc.) are stored in the form of XML files in Smart Bookmarks (SMBs). The selected page content is specified as XPath expressions. Such expressions are created automatically by using a point-and-click interface. This interface generates an XPath syntax from content marked by the user. The user can then access the chosen Web View via an URL on a mobile phone, a PDA, or through voice interfaces. The whole architecture of WebViews, depicted in figure 6.9, consists of a server, separate proxies for different markups, and various clients.

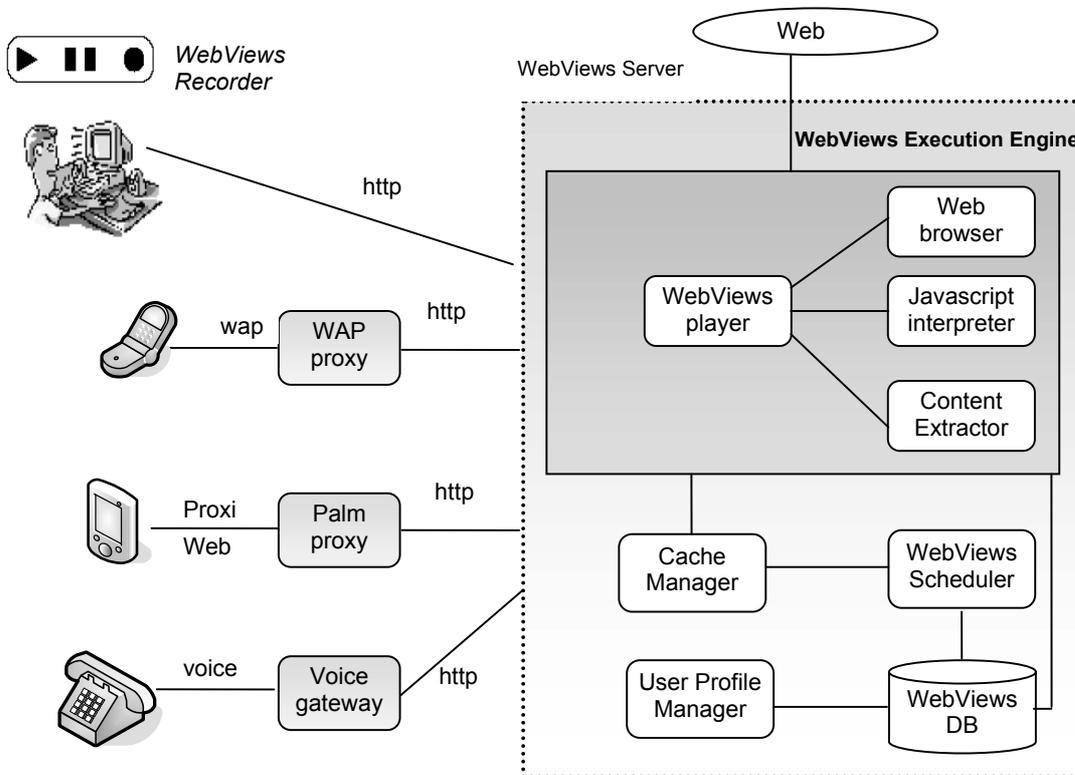


Figure 6.9: WebViews architecture [FrKuLi01, p. 578]

A User Profile Manager, a Scheduler, a Cache Manager, and an Execution Engine reside on the WebViews server together with a database. The Profile Manager performs users' authentication, the Scheduler executes Web views and the Cache Manager caches them. The Execution Engine in the interaction with the WebViews player, Web browser, and JavaScript interpreter retrieves, parses, and corrects HTML sites. The execution process in WebView is performed as follows: the user sends a request to the server, the recorded SMB is replayed, the final page is retrieved, and the chosen parts of the page are extracted by an XSLT processor with the help of XPath expressions. The selected content is subsequently sent to the appropriate proxy that transforms page fragments to the markup language supported by the client that requested the view. The transcoding process is simplified to a great extent because only relevant parts of the page have to be transduced. However, WebViews is not able to locate views correctly if pages change very often. In such cases, error messages are displayed and the user has to re-record his/her views. Furthermore, this method is only applicable if the user is willing to identify interesting content using a browser and the WebViews recorder on a desktop computer.

6.2.8 Opera Mini

Opera Mini⁸⁰ is a Java ME-based Web browser that needs only 50-100 KB and uses a special Opera Mini server with the Small- Screen Rendering technology on it. When the user requests a page, it is first pre-processed on the Web server with the help of SSR. Web pages are compressed by 70 to 80 percent and are then sent to the Java ME application that displays them. Figure 6.10 shows an example of an HTML page (<http://java.sun.com>) displayed in the Opera Mini browser. A demo of the browser is provided under the address: <http://www.opera.com/products/mobile/operamini/demo.dml>. Since no pagination is used, the user needs to scroll excessively to reach the end of the page. Similarly to the Opera Web browser, the content was squeezed to fit on the small display and images were resized. This approach allows seeing traditional Web sites on mobile devices but it is not user-friendly. The displayed sample page possesses, for example, a quite complicated navigation structure with a large number of links. If the user is completely unfamiliar with the Sun's Java page, he/she will certainly have serious difficulties to find relevant information.

⁸⁰ Opera Mini can be downloaded from <http://www.opera.com/products/mobile/operamini/phones/> with the help of a browser on a desktop computer or, alternatively, via OTA.



Figure 6.10: Sun's Java page displayed in Opera Mini browser

6.3 Approaches for server-side adaptation

With the proliferation of mobile devices, server-side adaptation approaches have gained increasing recognition. Such adaptation methods do not aim at squeezing existing Web content to the sizes of mobile screens or at the conversion to appropriate markup languages. Most server-side adaptation approaches are based on novel meta-languages from which a device-specific markup is dynamically generated on the server. In server-side adaptation three general categories of approaches can be distinguished: User Interface Description Languages, component-based architectures such as frameworks relying on the Model-View-Controller design pattern, and XML/XSLT-based methods.

6.3.1 General technologies for server-side adaptation

Several Java technologies are used across most of the open-source, server-side adaptation approaches. Java Servlets, JavaServer Pages, JSP Tag Libraries, JavaBeans, and Enterprise JavaBeans that are the most popular Java components, are described in this section.

Java Servlets

Java Servlets, released in version 2.4, are a Java technology that provides a component-based, platform-independent method for building dynamic, Web-based applications [Hall01; Sun04]. A servlet is a Java class compiled into byte code. It has access to a rich API of HTTP-specific services and is usually plugged into a container attached to a Web server (e.g. Tomcat, JBoss, etc.). Servlet handles predefined URI patterns. When a request matches a registered URI pattern, a Web server passes the request to a container, and the container invokes the servlet. Separate servlets are not created for

each request - instead of this, a new thread is generated for each request. Additionally, servlets can hold references to different resources (e.g. database connections) that are shared between requests and therefore increase application performance.

JavaServer Pages

JavaServer Pages (JSP), currently in version 2.1, are based on servlets and help to develop Web pages that include dynamic content [cf. Berg02; FiKoBa02; Patz02; Sun05a]. A JSP page can contain markup language elements, such as HTML or WML tags, special JSP elements that allow a server to insert dynamic content into a page and even a regular Java code. When a user sends a request for a JSP page, the server executes the available JSP elements, merges the results with the static parts of the page, and sends the dynamically composed page back to the browser. JSP pages need a JSP container for processing. The container is in charge of request interception for JSP pages. To process all JSP elements in the page, the container first converts the JSP page into a servlet and then compiles the servlet class. The JSP container invokes the JSP page implementation class (the generated servlet) to process each request and it produces the response.

JavaBeans

JavaBeans are Java classes that follow certain coding conventions and can therefore be used as software components in applications [Engl97]. A software component is defined as a reusable unit of deployment and composition, accessible through an interface [CrLa02, p. 5]. Components encapsulate data and hide implementation. To qualify as a JavaBean, a class has to be implemented as concrete and public and has to possess a constructor without parameters. Internal fields of JavaBeans are exposed as properties and can be accessed by using public methods that follow a certain design pattern (e.g. setter and getter methods have to be provided). An application can query a component through introspection⁸¹. The program can find out the capabilities of the component and interact with it.

Enterprise JavaBeans

Enterprise JavaBeans are server-side components in Java and can be deployed in a multi-tier environment [cf. Roma+05; Sun04b; HaBuLa04]. They are designed to accomplish server-side operations such as accessing databases or other systems or performing business logic. The communication with an enterprise bean is done with the help of the interface exposed by it. Three types of beans can be distinguished: Session Beans, Entity Beans, and Message-driven Beans. Session Beans model business processes, Entity beans are responsible for business data. Message-driven Beans are similar to Session beans, but are called by sending a message to them [Roma+05, pp. 27-30]. Enterprise JavaBeans are deployable and live in a container.

⁸¹ Introspection is a mechanism that exposes the functionality of components externally.

JSP Tag Libraries

JSP Tag Libraries [Sun03a] are reusable modules that are able to create and access JPS programming language objects and affect the output stream⁸². Tag libraries enhance the pre-programmed actions of JSP pages and include recurring tasks like XML or form processing. They can be reused across many applications and help to divide work between Java developers, responsible for coding of tag libraries, and Web pages designers. The most popular set of tag libraries is called JSP Standard Tag Library (JSTL) [Baye03]. It consists of libraries for XML processing, internationalization, database access, core actions such as iterations, conditional processing, and expression language support. Programmers can develop their own tag libraries and make them available for public use⁸³.

6.3.2 XML/XSLT-based architectures

The rapid growth of the traditional Internet has been stimulated by the fact that electronic documents, based on a standardized, open format, could be easily and inexpensively presented to a worldwide audience. This popular format, called Hypertext Markup Language (HTML), was a descendant of Standard Generalized Markup Language (SGML) published as ISO 8879 in 1986 [ISO86]. SGML has provided an arbitrary structure, data validation, and extensibility but was too difficult and costly to be implemented just for a Web browser⁸⁴. HTML, with its strict semantics and structure, fitted better to users' needs, and has functioned well as a markup for the publication of simple documents as well as a transportation envelope for downloadable scripts.

However, as Web documents have become larger and more complex, Web content providers have begun to experience the limitations of the format that does not provide the SGML characteristics. To address the requirements of large-scale commercial publishing and to enable further expansion of Web technology into new domains of distributed document processing, the World Wide Web Consortium (W3C) has developed Extensible Markup Language (XML).

The emergence of XML has exerted unexpected influence on mobile Internet; suddenly everybody became interested in eXtensible Markup Language. Markups for wireless devices were created basing on XML, information about device context was stored in XML files, and content adaptation to various mobile devices was achieved by separating content and presentation rules into XML and XSLT files. This section introduces the so-called family of XML languages. It briefly describes some approaches for content adaptation based on the processing of XML files and transformation of this data format with the help of XSL.

⁸² More information on tag libraries can be found in section 8.3.2.

⁸³ A sample collection of tag libraries can be found at <http://jsptags.com/index.jsp>.

⁸⁴ SGML consists of many optional features. Both the sender and the receiver of electronic documents have to agree on some set of options. SGML systems are able to solve large, complex problems that justify their expense. Viewing structured documents, sent over the Web, rarely carries such justification.

6.3.2.1 Underlying standards

The most important standards associated with XML are DTD, XMLSchema, XPath, XPointer, XLink, and XSL. In DTD or XMLSchema, the developer can specify the structure of an XML document and define a list of legal XML elements. XPath and XPointer are declarative languages for addressing XML nodes or fragments, XLink can be used to link XML resources. XSL is a set of languages that can transform XML into various markup languages, filter and sort XML data. Moreover, it can output XML to different media like screens, paper, or voice.

eXtensible Markup Language (XML)

eXtensible Markup Language (XML) [W3C04b] is a meta-language defined by the World Wide Web Consortium. It is used to describe a broad range of hierarchical markup languages such as VoiceXML, XHTML, SVG, WML, or SMIL. XML does not specify any particular elements of languages; instead of this, it provides a set of rules, guidelines, and conventions for presenting structured data.

```
1. <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
2. <?xml-stylesheet type="text/xsl" href="author.xsl"?>
3. <!DOCTYPE author SYSTEM "C:\author.dtd">
4. <!--XML comment-->
5. <authors>
6.   <person id="1">
7.     <name>Jankowska</name>
8.     <firstName>Bozena</firstName>
9.     <age>28</age>
10.  </person>
11.  <person id="2">
12.    [...]
13.  </person>
14. </authors>
```

Listing 6.3: Sample XML document (*author.xml*)

XML documents may consist of different syntactic constructs as presented in listing 6.3: elements (e.g. *authors*, *person*), declarations of namespaces, DTD declarations (line 3), attributes (*id* in line 6), processing instructions (line 2), comments (line 4), and text. XML documents have to be well-formed and conform to the rules determined in the XML specification⁸⁵. In order to make the data usable for different applications and to ensure their validation, each XML document should possess a separate definition of the data structure. The structure, against which the data is compared, can be provided in a number of different ways, but the most popular standards are Document Type Definition (DTD) and W3C XML Schema. Validation is not a requirement when working with XML data. The so-called valid

⁸⁵ According to the XML specification, XML element names are case-sensitive, must start with a letter or underscore, and cannot contain embedded spaces. The XML tags cannot overlap and end tags cannot be left out. Tags that do not enclose any text can contain the end marker at the end of the start tag. `<emptyTag/>` is therefore equivalent to `<emptyTag> </emptyTag>`. For more details, cf. [W3C04b].

documents follow rules of XML and its DTD or XML Schema. Although an XML document can possess a description of its structure, no additional semantic information is provided about XML tags. Therefore, only a narrow group of document creators or processors is able to understand the meaning of the XML document.

Document Type Definition (DTD)

The purpose of Document Type Definition (DTD) [W3C02e] is to provide the syntax for describing and constraining the logical structure of an XML document. DTD specifies the allowed elements and their values, indicates permitted elements' attributes, and lists the number of occurrences and the order of elements' appearance. A sample DTD is provided in listing 6.4. It defines all XML elements such as `authors`, `person`, or `name` and attributes (e.g. the `id` attribute for `person` element). Furthermore, it specifies the arrangement of elements (e.g. each `person` element should consist of the `name`, `firstName`, and `age` elements) and provides information about the number of occurrences of certain elements – the `authors` element should, for example, consist of one or more `person` elements.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!ELEMENT age (#PCDATA)>
3. <!ELEMENT person (name, firstName, age)>
4. <!ELEMENT authors (person+)>
5. <!ELEMENT firstName (#PCDATA)>
6. <!ELEMENT name (#PCDATA)>
7. <!ATTLIST person CDATA #REQUIRED>
```

Listing 6.4: DTD for `author.xml` document (`author.dtd`)

DTD has severe shortcomings: it cannot restrict the content of elements and is not able to express complex relationships. In DTD, the content of an element can be constrained to a string value (`#PCDATA`) or child elements. Customized data types or data types different from strings are not possible. Since DTD is not hierarchical, it enforces new names for similar elements in different settings because the names cannot be repeated. DTDs do not permit to formally specify field-validation criteria such as the length of a string or the valid interval for values. Moreover, a DTD uses a syntax that is substantially different from XML and cannot therefore be processed by a standard XML parser.

XML Schema

XML Schema [Vlis02; W3C01c] is an alternative to DTD. It is based on the XML syntax and is extensible. DTD defines XML elements and attributes, assigns specific data types to the elements or attributes, and restricts their values (for example with the help of regular expressions). Data types can refer to custom types (e.g. derived from the existing data types) or to built-in types listed in the XML Schema recommendation⁸⁶. Adding children and attributes to an element requires the use of complex

⁸⁶ 42 simple types are defined as part of the recommendation, including `string`, `int`, `date`, `decimal`, `boolean`, `timeDuration`, and `uriReference`, cf. [W3C01c].

types (`<xs:complexType>`) instead of simple types. XML Schema offers better content modeling than DTD. It provides the ability to constrain the order and number of child elements, to define a minimum and maximum number of consecutive instances of an element (`minOccurs/maxOccurs`), and to define groups of elements without specifying their order. Listing 6.3. shows a sample XML Schema for the XML document *author.xml* introduced in the previous section. Compared with the DTD for the same document, XML Schema features a much higher complexity and extensibility.

```
1. <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3.   elementFormDefault="qualified">
4.   <xs:element name="age">
5.     <xs:simpleType>
6.       <xs:restriction base="xs:string"/>
7.     </xs:simpleType>
8.   </xs:element>
9.   <xs:element name="person">
10.    <xs:complexType>
11.      <xs:sequence>
12.        <xs:element ref="name"/>
13.        <xs:element ref="firstName"/>
14.        <xs:element ref="age"/>
15.      </xs:sequence>
16.      <xs:attribute name="id" use="required">
17.        <xs:simpleType>
18.          <xs:restriction base="xs:string"/>
19.        </xs:simpleType>
20.      </xs:attribute>
21.    </xs:complexType>
22.  </xs:element>
23.  <xs:element name="authors">
24.    <xs:complexType>
25.      <xs:sequence>
26.        <xs:element ref="person" maxOccurs="unbounded"/>
27.      </xs:sequence>
28.    </xs:complexType>
29.  </xs:element>
30.  <xs:element name="firstName">
31.    <xs:simpleType>
32.      <xs:restriction base="xs:string"/>
33.    </xs:simpleType>
34.  </xs:element>
35.  <xs:element name="name">
36.    <xs:simpleType>
37.      <xs:restriction base="xs:string"/>
38.    </xs:simpleType>
39.  </xs:element>
40. </xs:schema>
```

Listing 6.5: XML Schema for *author.xml* document

eXtensible Stylesheet Language and XML Path Language (XSL and XPath)

The eXtensible Stylesheet Language (XSL) [W3C03e] standardization effort started out with the ambitious goal to normalize the styling and rendering of content. The standard was divided into two parts: XSL Transformations (XSLT) for transforming document's structure and XSL Flow Objects (XSL-FO) for precisely specifying the layout. XSL is complementary to CSS because it does not simply provide formatting and styling functionality, but is also able to change the content and structure of a transformed document.

A transformation expressed in XSLT describes the rules for converting an input tree into an output tree. The conversion is achieved by associating patterns with templates. Whenever a pattern matches elements in the source tree, a template is used to create a part of the result tree. The result tree is different from the source tree and their structures can vary completely. In constructing the result tree, elements from the source tree can be filtered as well as reordered, and new elements can be added.

XSLT is a combination of three basic components: a pattern matcher, a rules engine, and a template processor. The pattern matcher has the form of `<xsl:template match="/">`. This tells the XSL processor to apply the subsequent template when it finds a match to the matching argument, in this case the document root. The rules engine resolves conflicts and decides which pattern should take precedence if multiple patterns can be applied. The template processor executes a set of instructions as a consequence of the pattern match. The instructions create new elements in the output tree, either implicitly by using non-XSL (e.g. HTML tags) or explicitly using the `<xsl:element>`, `<xsl:attribute>`, or `<xsl:text>` tags. Finally, the `<xsl:apply-templates>` tag hands the control over to the pattern matcher that searches for the next rule to be executed.

XML elements are addressed using XML Path Language (XPath) [W3C04j]. The syntax of XPath has a strong resemblance to directory path specifications. XPath provides basic facilities for the manipulation of strings, numbers, and boolean values. Location paths help to select one node or a set of nodes in the document tree. XPath supports various functions, for instance, node-set functions (e.g. `count(node-set)` for counting the number of nodes), sub-string and concatenation functions (e.g. `substring(string, number)` for retrieving a specified number of characters from a string), boolean functions (`not(boolean)` for logical negation), and functions for numerical calculations (e.g. `sum()`). It is therefore possible to form complex matching expressions.

A sample XSLT stylesheet for transforming and displaying the "author.xml" document is presented in listing 6.6. The stylesheet converts XML into HTML, adds formatting and styling to the elements, and displays them as a table with a header, presenting the authors.

A common server-side adaptation method consists of retrieving data from an information system in the XML format and converting them on the server-side to the appropriate markup language with XSLT (cf. also figure 6.11) and an XSLT processor. Different processors, implemented in various programming

languages, exist. Popular Java processors are Xalan-Java from Apache⁸⁷ and the Transformation API for XML (TrAX) from Sun⁸⁸. In this technique, content is separated from presentation and the same data can be presented in different ways, depending on the style sheet. However, if the view changes, every style sheet has to be updated separately.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3. xmlns:xs="http://www.w3.org/2001/XMLSchema"
4. xmlns:fn="http://www.w3.org/2004/07/xpath-functions"
5. xmlns:xdt="http://www.w3.org/2004/07/xpath-datatypes">
6. <xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>
7.   <xsl:template match="/">
8.     <html>
9.       <body style="background-color:#EEEEEE;">
10.        <h3> The list of our authors:
11.        <i>(<xsl:value-of select="count(authors/person)"/> authors)</i>
12.        </h3>
13.        <table border="1" width="100%" style="background-
14.          color:whitesmoke;color:black;">
15.          <tr style="background-color:gray;">
16.            <td>First name</td><td>Name</td><td>Age</td>
17.          </tr>
18.          <xsl:for-each select="authors/person" xml:space="preserve">
19.            <xsl:call-template name="author-summary"/>
20.          </xsl:for-each>
21.        </table>
22.      </body>
23.    </html>
24.  </xsl:template>
25.  <xsl:template name="author-summary">
26.    <tr><td width="30%">
27.      <xsl:value-of select="firstName"/>
28.    </td><td width="40%">
29.      <xsl:value-of select="name"/>
30.    </td><td width="20%">
31.      <xsl:value-of select="age"/>
32.    </td></tr>
33.  </xsl:template>
34. </xsl:stylesheet>
```

Listing 6.6: XSLT stylesheet for author.xml

⁸⁷ Cf. <http://xml.apache.org/xalan-j/>.

⁸⁸ <http://java.sun.com/xml/jaxp/index.jsp>.

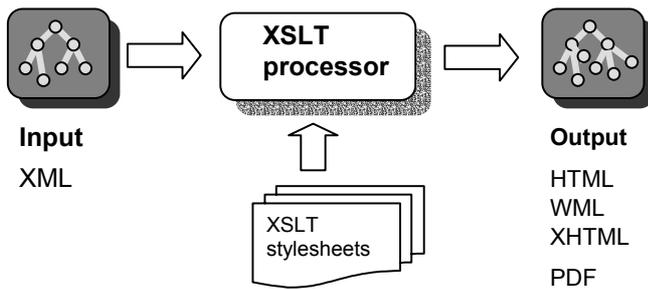


Figure 6.11: Converting XML data to different output formats

XML Pointer Language and XML Linking Language (XPointer, XLink)

The linking model, underlying HTML and the Web, is very simple. It is based on static, directional, single-source, single-destination links⁸⁹ that connect two pieces of information. Dynamic links, whose structure or behavior change over the time, can be created with the help of server-side technologies such as servlets or JSP. With the emergence of XML, a more sophisticated model consisting of XML Linking Language (XLink) and XML Pointer Language (XPointer) [W3C02f] was provided.

XML Pointer Language (XPointer) extends the XPath specification to support inter-document addressing and specifies the syntax for using fragment identifiers with XML resources. Links can therefore refer to specified parts of a document and can change dynamically. XPointer helps to specify links that are defined by a pattern to be matched to the content instead of a location (cf. listing 6.7, lines 3, 5). XLink allows for the definition of links as associations between a number of resources. In XLink, the concept of relations resources was separated from the concept of traversal between resources. The traversal can be realized with the help of arcs. An arc can start from multiple, different locations and can end on many resources. Listing 6.7 shows a multi-ended link, built dynamically using XPointer. XPointer selects the location of parents and children elements, and the arc specifies the connections between the resources: the user can navigate from different parents locations to various children locations.

```

1. <family xlink:type="extended">
2.     <loc xlink:type="locator" xlink:label="parents"
3.         xlink:href="Family.xml#xpointer(//Person[@type='parent'])"/>
4.     <loc xlink:type="locator" xlink:label="children"
5.         xlink:href="Family.xml#xpointer(//Person[@type='child'])"/>
6.     <go xlink:type="arc" xlink:from="parents" xlink:to="children"/>
7. </family>
  
```

Listing 6.7: XLink/XPointer example

⁸⁹ Static means that a link does not change over the course of time, directional refers to the fact that a link possesses an explicit direction of association and single-source, single-destination indicates that a client traverses a link from one source to one destination, cf. [WiLo02].

6.3.2.2 *Exemplary architectures based on XML/XSLT*

Architectures built upon the XML/XSLT standards prevail nowadays due to the lack of a standard markup language that would be able to support multiple wireless devices. Additionally, most languages for mobile devices (e.g. XHTML, VoiceXML, WML) were derived from XML what facilitates the transformation from pure XML to the desired end-format. In such frameworks, XML is used to handle data and XSLT is applied to extract and format data for presentation in different markups. The most popular XML/XSLT approach is Apache Cocoon. This section introduces this framework as well as some projects using the XML/XSLT set of technologies.

EIHA?!?

EIHA?!? is a project aiming at indexing, classification, and integration of Web sites about Sardinia [Bian+01]. The information can be presented in the HTML, WML, or SVG format. A collection of about 1700 resources about Sardinia was stored in a database and can be retrieved in the form of XML, conforming to specified DTDs. Java servlets are responsible for the retrieval process and for the recognition of users' devices. They perform the conversion from XML to the appropriate markup language with the help of XSLT style sheets. The process looks as follows: a servlet accepts the requests from a client and detects the type of device. It looks for an appropriate XSLT style sheet and applies it to the XML content. The result of the transformation is sent as a response to the client. This kind of request processing is characteristic for most of the architectures based on XML/XSLT [cf. DaGrJo05; ScKo03]. In some implementations, Java servlets are replaced by JavaServer Pages [cf. KuJaDa+03].

Apache Cocoon

Cocoon [BrCrGa02; Mazz01; MoAs02] is an open-source publishing framework initially developed by Stefano Mazzocchi as a Java servlet that provides content in multiple languages and formats such as HTML, XHTML, WML, SVG, or PDF. It is based on the concept of separation of concerns (SoC) and component-based Web development. Four major concerns in the area of Web publishing are distinguished and separated in the Cocoon framework: management, logic, style, and content. This separation results from the observation that sorting out people with common skills in different working groups increases productivity and reduces management costs, but only if the tasks of groups do not overlap.

Competencies are divided by using "component pipelines", where each component in the pipeline specializes in a particular operation. Cocoon recognizes three types of pipeline components: generators, transformers, and serializers. Generators are responsible for accepting requests and creating XML structures from input sources. The output produced by a generator is converted into another XML structure by using a transformer or a set of transformers. The most commonly used transformer is the XSLT transformer. Serializers handle the task of producing responses in desired formats. Serializers do not always need to be invoked from a transformer; a generator could also directly invoke a serializer. SAX events are passed between generators, transformers, and serializers (cf. figure 6.12).

Additional crucial components of Cocoon are a sitemap and matchers. The sitemap mainly consists of declarations for pipelines, components, and resources. It enables Cocoon to decipher how to tackle a particular request or how to find a resource. Matchers are sitemap components that are used to identify what request has come in and what resource is to be utilized. Using them, the sitemap is able to perform matching.

The Cocoon model allows sites to be highly structured and well-designed, reduces the duplication of effort and decreases management costs. It permits different presentations of the same data, depending on the requesting client. The main disadvantage of this framework is the slow transformation of XML files into the required markup language and the lack of unified Web identifiers for each markup language.

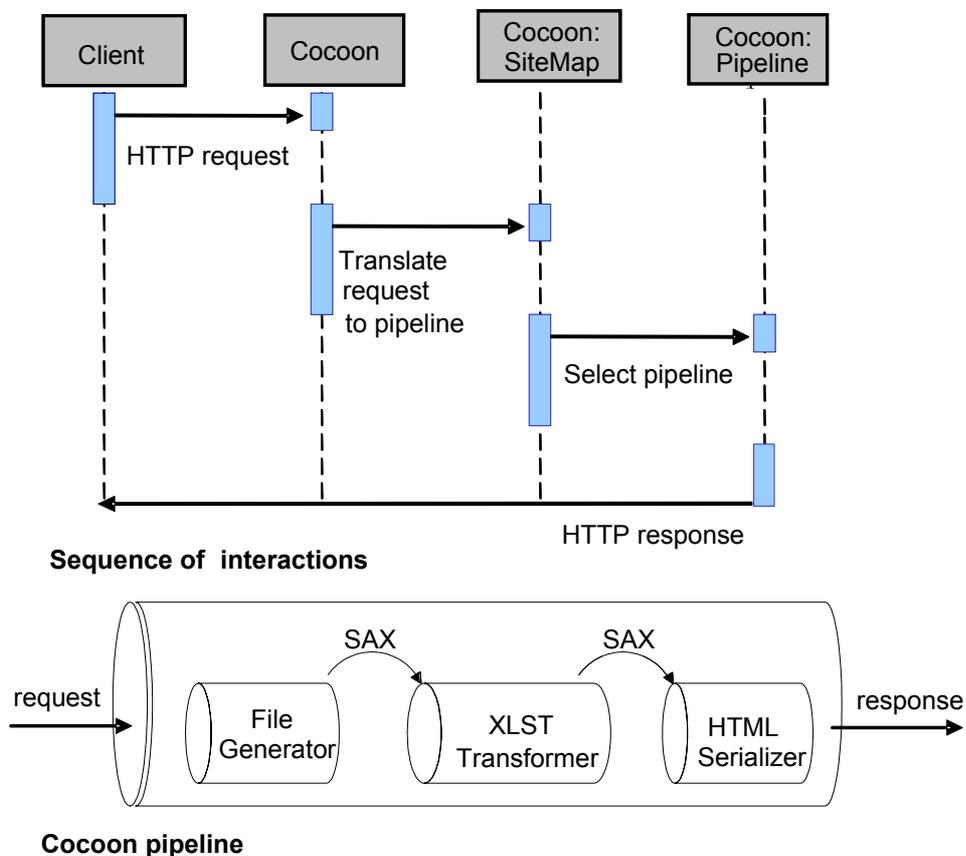


Figure 6.12: Request processing in Cocoon

Dynamic generation of XSLT style sheets

In XML/XSLT-based frameworks, separate style sheets have to be used to generate suitable presentations for various devices. A developer has to construct manually $N \times M \times L$ style sheets for an application designed for N devices, using M XML documents and generating L decks for each XML document. To reduce efforts and the amount of time invested in the development and maintenance of multiple style sheets, Kwok et al. propose to use a GUI tool that would automatically generate XSLT style sheets basing on the required presentation [Kwok+04].

The method consists of the generation of two separate sets of rules – one for content extraction and another one for formatting (cf. figure 6.13). These sets can be entered directly by the user or can be selected with the help of a Graphical User Interface. An XSLT style sheet is created in the setup stage by the so-called XSLT Style Sheet Generator that merges the information about the content and presentation. Subsequently, the style sheet is stored in a pool of style sheets for the runtime use. A suitable style sheet is selected by the XSLT Style Sheet Selector depending on device capabilities. After the conversion of XML files to the appropriate end formats such as WML, cHTML, HDML, or VoiceXML, the content is delivered to various wireless devices. The main advantage of automatic generation of XSLT style sheets is the uncomplicated portability of content to a variety of devices – a developer does not have to know the details of different languages and the efficiency of creation and maintenance of different presentations can be essentially improved.

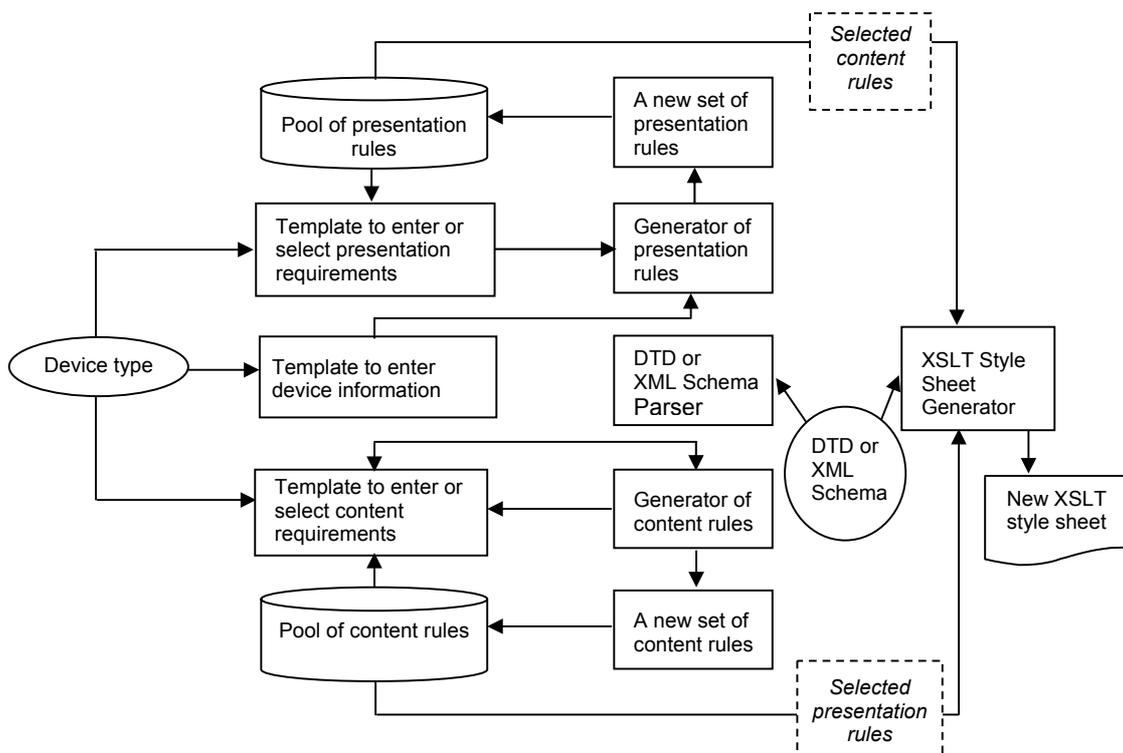


Figure 6.13: Dynamic generation of XSLT style sheets [Kwok+04, p. 218]

6.3.3 Approaches based on User Interface Description Languages

Declarative User Interface Description Languages (UIDLs) and Model-based User Interface Development existed for about a decade, mainly in the academic research [cf. StRu98; Szek+95; Szek96]. With the proliferation of heterogeneous devices, these concepts were revived and applied to many approaches for device-independent content adaptation. This section introduces the notion of Model-based User Interface Development and presents some XML-based User Interface Description Languages that provide device-independent, abstract definitions of user interfaces.

6.3.3.1 *Model-based User Interface Development and UIDLs*

The development of applications targeted at multiple devices implies a considerable effort because different presentations have to be implemented for displaying the same or similar data. In automated content adaptation approaches, the elements of a device-independent description language have to be mapped to appropriate building blocks of a device-specific description language (e.g. XHTML). The mapping can follow the intersection approach or the generic language approach. In the first case, the characteristics of a device-independent markup language are restricted to the common features supported by all devices. Since the elements of this device-independent language are translated into their device-specific counterparts, the device-independent language has to change if device-specific markups are modified. Elements that are device-specific and cannot be applied to all devices are not supported in the intersection approach. As an alternative to the intersection approach, a generic device independent format can encompass characteristics not featured by all device-specific markups. The mapping to specific representations may be loose and the features of the generic language are not restricted to those applied in a device-specific language with the “lowest common denominator” capabilities [cf. Goeb+01].

In the UI development paradigm known as Model-based User Interface Development, a generic interface using a high-level specification language is constructed. A generic user interface is “an interface whose aspects may vary in different devices while its functionality prevails in any of them” [Mayo02, p. 2]. The interface is automatically rendered according to device characteristics. Such high-level languages are called User Interface Description Languages (UIDL) and express diverse aspects of user interfaces, including their abstract and concrete elements, the tasks to be performed by the user, and the user interface dialogues.

User Interface Description Languages are based on declarative models. A declarative model is defined as “a common representation that tools can reason about, enabling the construction of tools that automate various aspects of interface design, that assist system builders in the creation of the model, that automatically provide context-sensitive help and other runtime assistance to users” [Szek+95, p. 120].

Model-based User Interface Development (MUID) relies on a set of models. Well-known models include data/domain models, application models, task models, dialog models, presentation models, and user models. In Model-based User Interface Development, information is usually categorized into three levels of abstraction [cf. Szek96]. At the highest level task, domain, and user models are placed. A task model describes the tasks to be accomplished by the user, while a data or domain model provides a description of the objects the user manipulates and the supported operations. A user model provides information about a user or a group of users.

The second and third levels in MUID are responsible for presentation. The second level represents an abstract user interface and specifies the information that will be shown in windows and the dialogs to

interact with the information (dialog model). The abstract user interface consists of abstract interaction objects (AIOs), information elements, and presentation units⁹⁰. AIOs correspond to interface tasks such as selecting one element from a set or showing a presentation unit. Information elements represent the information to be shown in the form of constant values (e.g. label), or a set of objects and attributes drawn from the domain model. Presentation units are an abstraction of windows and describe a collection of AIOs and information elements displayed to a user as a presentation unit.

The third level in MUID, the concrete or final user interface specification, denotes a style for displaying the presentation units, the AIOs and information elements they include, and the layout of elements. It corresponds to the interface in terms of toolkit primitives such as windows, buttons or checkboxes, and graphical primitives such as lines or images.

UIDLs may have different levels of abstractions. The instance level means that the user interfaces are runnable, the model level signifies that one or many models are involved in the development of UIs. If a language specifies the models and their semantics, it is at the meta-model level. The meta-meta-model level is achieved if the fundamental concepts about meta-model development for the meta-model level interfaces are also provided [SoVa03, p. 387].

6.3.3.2 *Related work in the area of User Interface Description Languages*

This section presents the most meaningful approaches in the area of User Interface Description Languages. The description of all existing languages is out of the scope of this thesis because the number of new frameworks and UIDLs is continuously growing. Some interesting languages, that are not described here, include eXtensible User Interface Language (XUL) from Mozilla⁹¹, Alternate Abstract Interface Markup Language (AAIML) [ZiVaGi02], SEESCOA XML [Coni+04; LuCo01], and Abstract User Interface Markup Language (AUIML) from IBM [AzMeRo00]⁹².

User Interface Markup Language (UIML)

User Interface Markup Language (UIML) [AbPh99; Abr+99; Abra00; Abra00a; AlPeAb04] is one of the most popular approaches for delivering information to different devices in a device-independent way. It is an XML-based language that provides a declarative description of user interfaces and has the goal of specifying a canonical format for multiple devices. UIML was designed to separate user interface code from application logic code, to facilitate the reuse of code, and to make rapid prototyping of user interfaces for multiple devices possible. Currently, UIML is being standardized by the Organization for the Advancement of Structured Information Standards (OASIS).⁹³

⁹⁰ This classification is taken from [Szek96]. Other authors do not divide the abstract presentation model into specific components.

⁹¹ Cf. <http://www.mozilla.org/projects/xul/>.

⁹² For more UIDL please refer to [OASI04, Luyt+04].

⁹³ Cf. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uiml.

An UIML document can consist of seven main elements: `<interface>`, `<structure>`, `<content>`, `<behavior>`, `<style>`, `<template>`, and `<peers>`. The `<interface>` element embraces all other tags and represents an user interface. In the `<structure>` element, the physical organization of the interface and the relationships between UI elements within the interface are defined. The `<content>` elements enclose the content of a document (e.g. text, images) in XML tags that separate it from the UI structure. The `<behavior>` elements describe the behavior of the interface by specifying conditions (e.g. the occurrence of an event) and actions associated with them. The `<style>` elements specify the presentation style of UI elements and the `<peers>` elements associate widgets, methods, programs, or objects in the application logic with the user interface, combining application presentation with its logic. The `<template>` elements help to describe those parts of the UI that are reusable [cf. Harm02 for the complete specification]. The main components of UIML and their relationships are depicted in figure 6.14.

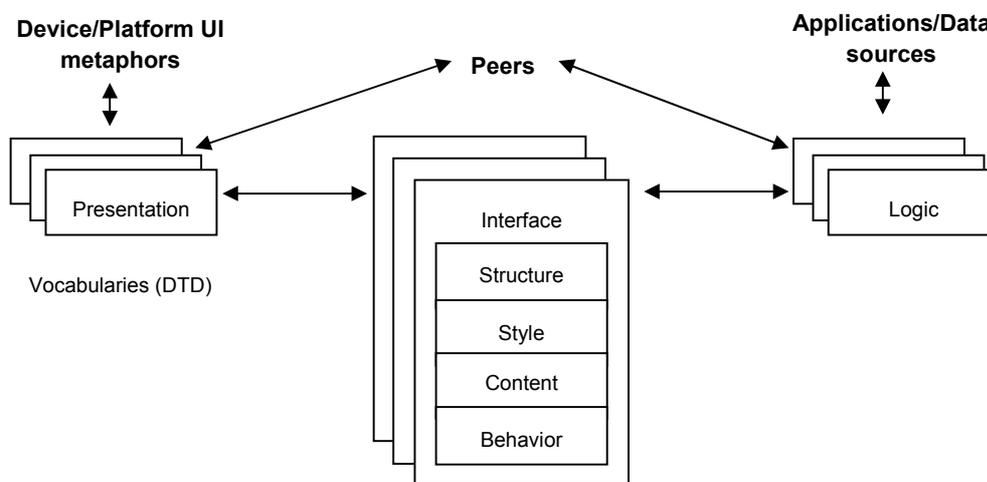


Figure 6.14: Meta-interface model of UIML [Harm02, p.14]

UIML documents can be mapped to any UI type (e.g. Java AWT, WML, VoiceXML, HTML) with the help of appropriate renderers⁹⁴. Listing 6.8 shows an UIML specification of the UI, which displays the “Hello World” text and is mapped in the `<peers>` component to WML. Listing 6.9 presents the rendered WML code.

Most of the UIML renderers are commercial pieces of software but the language specification is freely available for public use and can be extended with additional vocabularies. For example, in the Mobile multimodal Next-generation Applications (MONA) project, a specific vocabulary for multimodal interfaces and a suitable renderer for graphical and voice user interfaces were developed [Ane+04; Simo+04; Wegs+04]. Similarly, in the Multimodal Interaction and Rendering System (MIRS), the authors developed Dialog and Interface Specification Language (DISL). DISL is an UIML subset enhanced with rule-based descriptions of state-oriented dialogs for advanced multimodal interfaces [MuScBI04].

⁹⁴ Most of the renderers were developed by the Harmonia company, cf. <http://www.harmonia.com>.

```

1. <?xml version="1.0"?>
2. <!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
3. "http://uiml.org/dtds/UIML3_0a.dtd">
4. <uiml>
5.   <interface>
6.     <structure>
7.       <part id="TopHello">
8.         <part id="hello" class="helloC"/>
9.       </part>
10.    </structure>
11.    <style>
12.      <property part-name="TopHello" name="rendering">Container
13.    </property>
14.      <property part-name="TopHello" name="content">Hello
15.    </property>
16.      <property part-class="helloC" name="rendering">String
17.    </property>
18.      <property part-name="hello" name="content">Hello World!
19.    </property>
20.    </style>
21.  </interface>
22.  <peers>
23.    <presentation name="WML">
24.      <component name="Container" maps-to="wml:card">
25.        <attribute name="content" maps-to="wml:card.title"/>
26.      </component>
27.      <component name="String" maps-to="wml:p">
28.        <attribute name="content" maps-to="PCDATA"/>
29.      </component>
30.    </presentation>
31.  </peers>
32. </uiml>

```

Listing 6.8: Simple UIML document [Harm02, p. 16-17]

```

1. <?xml version="1.0"?>
2. <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.0//EN"
3. "http://www.wapforum.org/DTD/wml.xml">
4. <wml>
5.   <card title="Hello">
6.     <p>Hello World!</p>
7.   </card>
8. </wml>

```

Listing 6.9: WML code generated from UIML document [Harm02, p.17]

eXtensible Interface Markup Language (XIML)

eXtensible Interface Markup Language (XIML) is another UIDL developed to provide a common representation of multiple user interfaces [cf. EiVaPu00; EiVaPu+01; PuEi01; PuEi02]. The language is “an organized collection of interface elements that are categorized into one or more major interface components” [PuEi01, p. 3]. XIML contains components, relations, and attributes, whereby relations

and attributes can be in the form of statements or definitions (cf. figure 6.15). The following five basic components can be distinguished in the language specification: task, domain, user, presentation, and dialog. The task component supports a definition of business processes and user tasks. The domain component represents a collection of data objects and classes of objects structured in a hierarchy. The user component specifies a hierarchy of users. The presentation component defines a hierarchy of interface elements such as windows, buttons, labels, etc. The dialog component describes a collection of elements determining user actions associated with particular interface components. A relation in XIML is described as a definition or a statement that connects two or more elements within one component or across many components. Attributes are features or properties of elements.

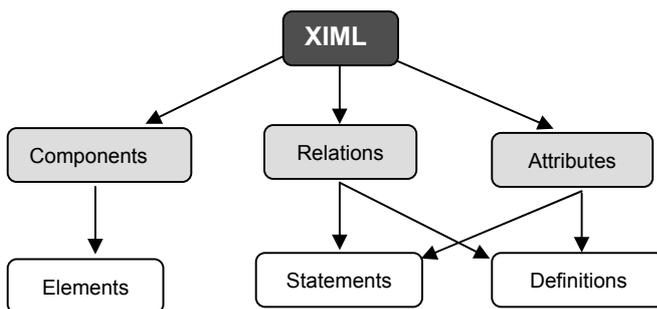


Figure 6.15: Basic structure of XIML [PuEi01, p. 3]

XIML was used in the Map Annotations Assistant (MANNA) project [EiVaPu00] for the creation of multiple user interfaces for annotated maps of geographical areas. In this work, some additional concepts for better adaptation to mobile devices were introduced: Abstract Interaction Objects (AIOs), Concrete Interaction Objects (CIOs), Logical Window (LW), and Presentation Unit (PU) (cf. figure 6.16). An interaction object (also called a widget) is any element that helps to visualize or manipulate information, or to perform a task [EiVaPu00]. AIOs are elements that cannot be executed on any platform and do not provide implementation details. CIOs are executable components and can be mapped to the platform on which they should run. CIOs are children of AIO; they inherit its properties and give information about implementation details. A Logical Window is a group of simple or composite AIOs (e.g. a window, a sub-window, a dialog box, a listbox) and is itself a composite AIO. A Presentation Unit is a complete presentation environment for enabling an interactive task and can consist of one or many Logical Windows, displayed simultaneously or one after another. This presentation hierarchy can be used for the generation of platform-specific presentations from platform independent presentation models.

The same concept was also applied in the reVerse engineering of Applications by Questions, Information and Transformation Alternatives (VAQUITA)⁹⁵ project [cf. BoVa02; BoVa03; BoVeCh04; SoVaBo01; VaFI04]. In this case, HTML pages were transformed to a presentation model with the help of reverse engineering. A HTML page was analyzed and specific mapping rules were applied to generate the model.

⁹⁵ Cf. <http://www.isys.ucl.ac.be/bchi/research/vaquita.htm>.

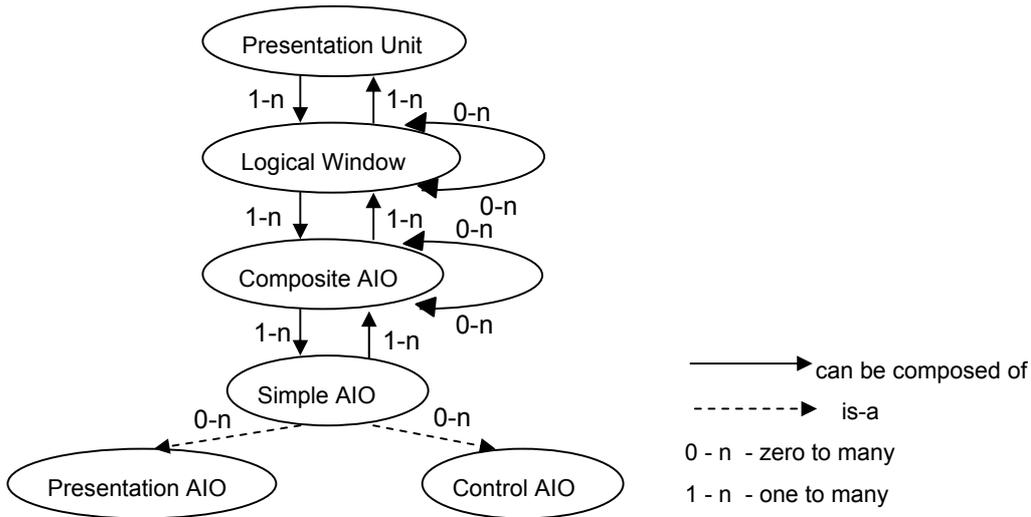


Figure 6.16: XIML elements [EiVaPu00, p. 88]

USer Interface eXtensible Markup Language (USIXML)

User Interface eXtensible Markup Language (USIXML) addresses multi-directional UI development and describes user interfaces with various levels of details and generalizations [cf. Limb+04; Vand+00; Vand+04]. The language was developed in the context of the Context Aware Modeling for Enabling and Leveraging Effective interaction (CAMELEON) project⁹⁶.

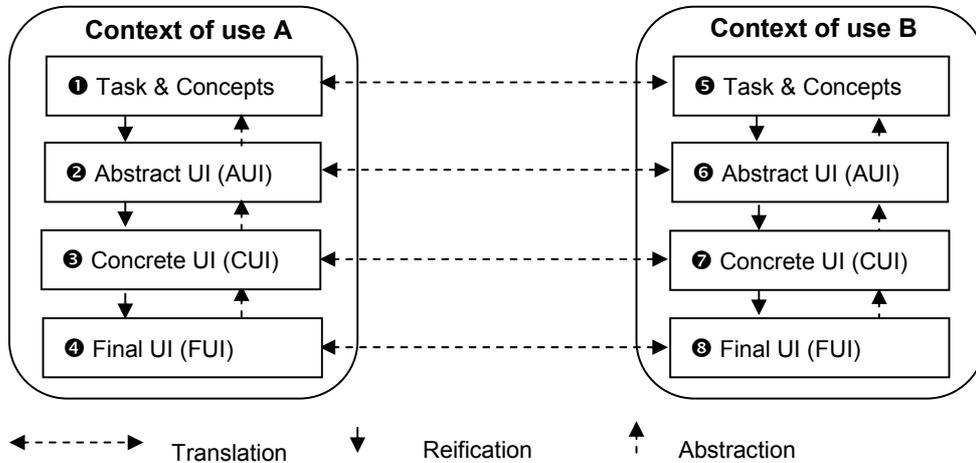


Figure 6.17: Cameleon reference framework (USIXML)

Four levels of abstractions, illustrated in figure 6.17, should express the UI development lifecycle for context-sensitive, interactive applications. A Final User Interface (FUI) corresponds to any UI running on a particular platform by execution (after compilation) or interpretation (e.g. in a Web browser). A

⁹⁶ Cf. <http://giove.cnuce.cnr.it/cameleon.html>.

Concrete User Interface (CUI) is a platform-independent UI definition. An Abstract User Interface (AUI) abstracts a CUI into a UI definition without any modality of interaction (e.g. graphical or vocal). At the last level, called Task and Concepts, interactive tasks and objects, manipulated by these tasks, are defined. The model partially resembles the XIML concept because it was developed by researchers who also participated in the development of XIML. USIXML possesses a collection of basic UI models – domain model, task model, AUI model, CUI model, context model as well as mapping model, and is also equipped with a transformation model specifying transformation steps and objects. The CUI and AUI refer to two levels of abstraction. The task model describes user tasks, the domain model specifies the classes of manipulated objects during interactions, the mapping model connects models or elements of models, the context model consists of a user model, a platform model, and an environment model, and therefore describes the context of the application's use. A set of tools was developed to support the edition of models in USIXML (the so-called GrafiXML tool), the reverse engineering of UI code (ReversiXML)⁹⁷ and the specification of transformation (TransformiXML).

MyXML and Device-Independent Web Engineering framework (DIWE)

Device-Independent Web Engineering (DIWE) framework [cf. KeKi00; Kere+01; KiKe00; Kird+01; Krue+01] was initially used for the generation of a multi-device Web presence of the famous Vienna International Festival. The framework consists of a specific, XML-based language called MyXML, a compiler for interpreting this language, and four default run-time processors. It aims at providing device-independent access to the Web and was designed to support all phases of the Web services engineering process: design, implementation, deployment, and maintenance. The framework separates layout, content, and business logic of applications. The separation is achieved by defining the content and its structure in MyXML documents, providing the layout separately in XSL style sheets, and isolating the business logic in the form of a particular programming language (e.g. Java).

In DIWE, MyXML documents encompass content enclosed in tags, predefined in a document type definition (DTD) file. The language possesses constructs such as loops, variables, arrays, parameters, and special functions for database access. Layout definition is provided in an XSL style sheet that can specify the way of processing the content for different devices. DIWE enables the creation of dynamic and static Web pages. Static pages are documents with tags that can be resolved during processing time. Dynamic pages contain tags that cannot be resolved at compilation. To this category belong runtime parameters, database queries, or user-defined variables.

If static content has to be produced, the MyXML language compiler processes the MyXML input document and the appropriate style sheet and generates HTML, WML, or XML pages from them (a typical XML/XSL processing). If dynamic content needs to be created, Java source code that handles dynamic aspects of a Web page, such as database queries or parameters passing, is produced. This code can then be used by servlets or JSP pages. Figure 6.18 illustrates the process of creating static and dynamic content in the DIWE framework.

⁹⁷ ReversiXML is available as a Web interface under <http://www.isys.ucl.ac.be/bchi/research/reversi/RevXMLUI.php>.

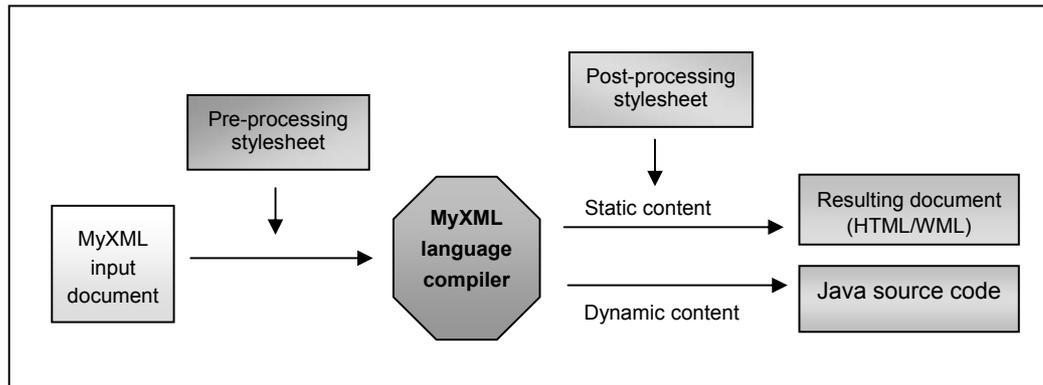


Figure 6.18: Simplified MyXML process [KeKi00, p. 140]

DIWE possesses four run-time processors: device-detection, logic interfacing, page splitting, and process partitioning processors. The device-detection processor detects different types of devices, the logic interfacing processor supports device-independent usage of the application logic. The last two processors are responsible for layout adaptation. The page splitting processor divides content into pages basing on the information about grouping elements and sub-elements, provided in MyXML documents or XSLT style sheets. Process partitioning processor helps to deal with form-based interactions on different devices. It is able to collect all form parameters before submitting them to a target URL even if the form is displayed on many screens [cf. Krue+01, pp. 51-52].

Additionally, DIWE was extended with the support for the creation, integration, and composition of SOAP-based Web Services [Kird+03]⁹⁸. An existing Web page can include an external Web Service (Web Services integration) or can be composed from two or more external Web Services (Web Services composition). The prototypical implementation was based on Apache SOAP toolkit, JLex, and JCup for the generation of code, and Apache Xerces and Xalan for XML processing [Kird+03, p. 285].

Dialog Description Language (DDL)

Dialog Description Language (DDL) is an XML-based, device-independent markup language that describes a structure of abstract elements [Buch+02; Goeb+01; Hueb+03; Hueb+05; SpGo02]. The root `<ddl>` element of a document may contain different elements like `<include>` for integrating external source code, `<DataTypeDef>` for the definition of data types used for the validation of user input, `<DataInstance>` for specifying data instances for user input, `<dialog>` for the definition of dialog structures and `<part>` for modeling the structure of a dialog. Furthermore, different classes can be assigned to parts. A class is defined as a set of `<properties>` (styles for presentation or a set of abstract properties). The content of DDL dialogs is enclosed in the `<content>` tags. A data item can be defined using `<constant>` elements. Figure 6.19 presents the DDL syntax.

⁹⁸ For more information on Web Services cf. Chapter 9.

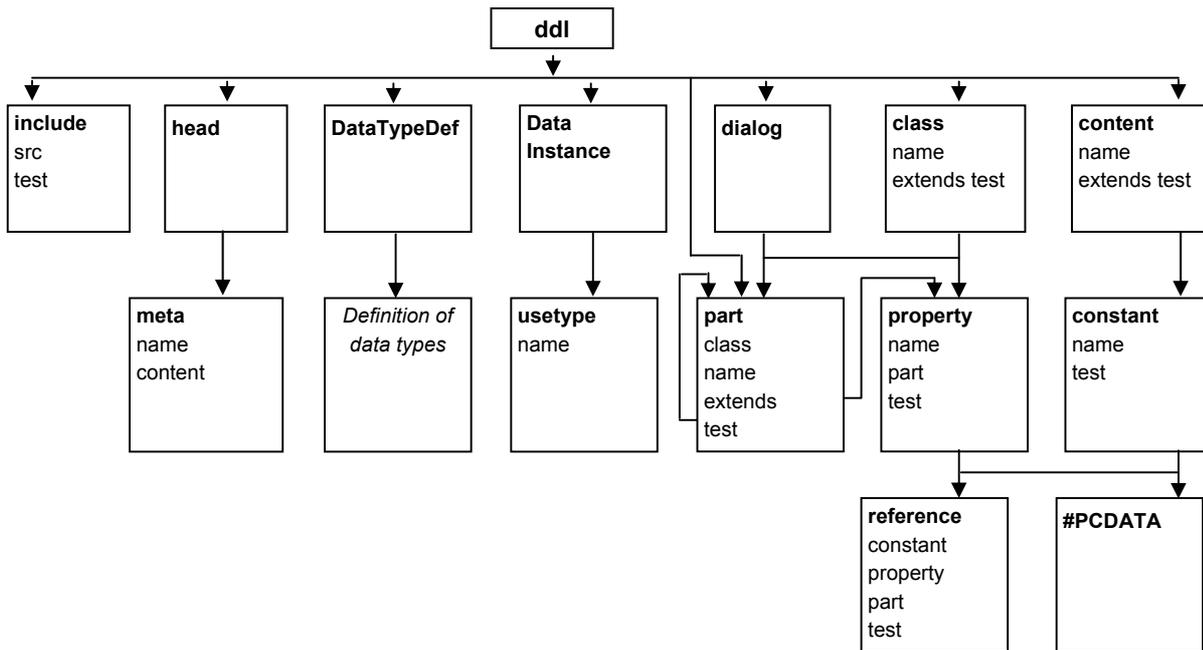


Figure 6.19: DDL syntax [Goeb+01]

The semantics of properties is defined separately in DTD and can be extended. The properties container and source element are particularly interesting; the remaining elements simply map to traditional Web-based UI elements - labels, textboxes, frames, forms, etc. The container enables the grouping of parts and specification of their layout. The source element enables the inclusion of a non-interpreted (not DDL) device-specific source code (e.g. WML).

In order to use DDL, an adaptation framework displayed in figure 6.20 was developed. The framework is based on a chain of filters that perform the adaptation according to the DDL specification. A filter is a Java class that can intercept a client request before it accesses a resource or a class that is able to intercept a response from resources before it is sent back to the client. The filter can also manipulate requests from clients or change responses before they are delivered to the client. Multiple filters can be applied to the same URL, building the so-called chain of filters⁹⁹. Three different types of filters were used in this adaptation approach: Request Modifiers, Generators, and Response Modifiers. Request Modifiers are processed at first and alter an HTTP request. Generators supply the requested content and Response Modifiers transform the retrieved content to the device characteristics. The sequence of filters is specified in a configuration file. Filter may also determine their successors.

⁹⁹ For more information on filters, cf. <http://java.sun.com/products/servlet/Filters.html>.

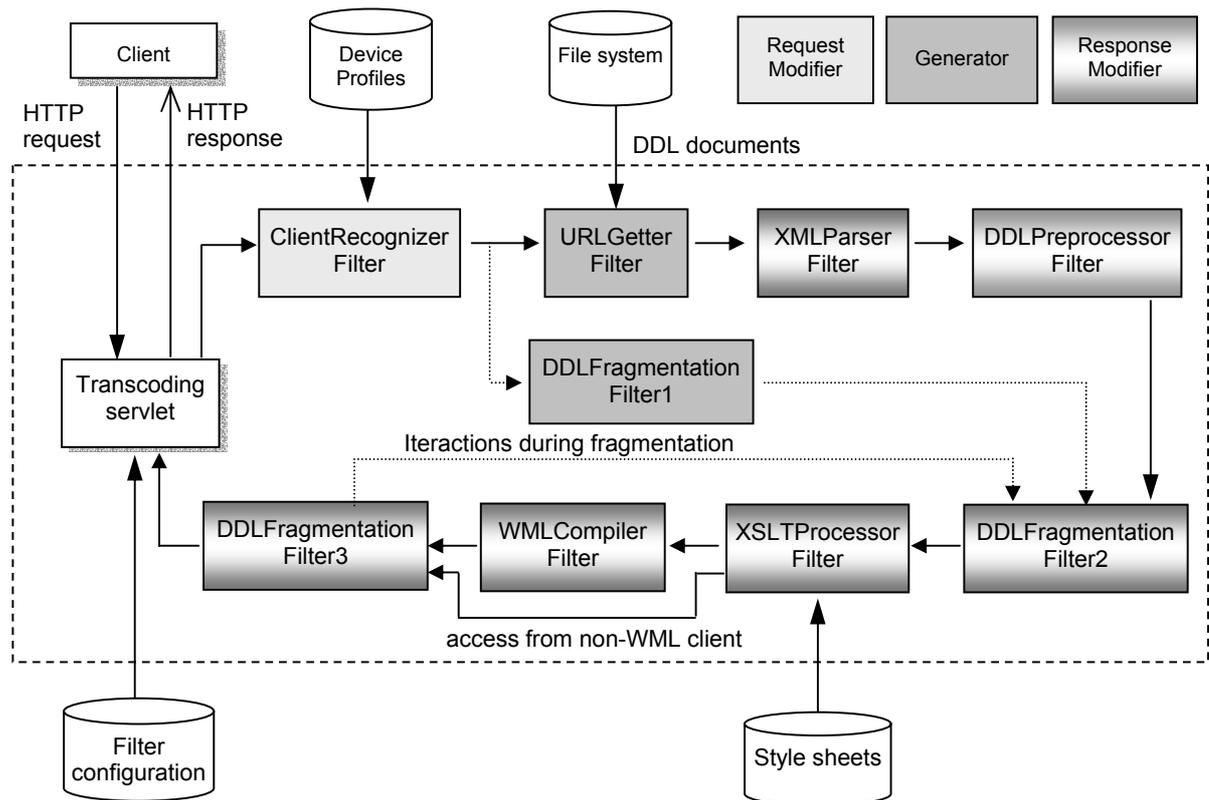


Figure 6.20: DDL adaptation framework [Buch+02, p. 47]

The most important filters implemented in this adaptation approach are: ClientRecognizerFilter, URLGetterFilter, ServletRunnerFilter, XMLParserFilter, DDLPreprocessorFilter, XSLTProcessorFilter, ImageTranscodingFilter, DDLFragmentationFilters, and WMLCompilerFilter. The ClientRecognizerFilter is responsible for the recognition of devices according to the User Agent String. The URLGetterFilter retrieves a file and the ServletRunnerFilter invokes an external servlet on the server. The XMLParserFilter converts a DDL document into a Document Object Model (DOM) instance, on which the subsequent filters work. The DDLPreprocessorFilter is in charge of resolving external references and inheritance hierarchies. It produces a simplified DDL document that is subsequently processed with the XSLTProcessorFilter using an XSLT style sheet to an appropriate end output. The ImageTranscodingFilter transforms images according to the device capabilities. The DDLFragmentationFilters fragment the dialogs, perform user input validation, and store input data. The WMLCompilerFilter compiles the textual representation of WML into binary format.

Consensus project and Renderer Independent Markup Language (RIML)

Renderer Independent Markup Language (RIML) is part of the CONSENSUS project (3G Mobile Context Sensitive Adaptability - User Friendly Mobile Work Place for Seamless Enterprise Applications) [cf. Derm+03; Gras+02; Spri+03; Zieg+04]. It aims at the development of highly usable mobile applications. RIML combines elements from already existing markups with new elements. Tags and concepts borrowed from XHTML 2.0, XForms 1.0, and SMIL can be found in this language [cf. Cons04]. All three of these markups are recommended by the W3C Consortium for device-independent applications [cf. W3C04; W3C03].

The structure of a RIML document is similar to an XHTML document. XHTML, as used in RIML, has some extensions, restrictions, and semantic modifications compared with the original XHTML. To address the needs of Web developers who build applications for a variety of devices, the W3C has invented XForms [W3C04i]. The XForms markup is characterized by the separation of content, presentation, and logic, similarly to Cocoon. Content is the fundamental data model, presentation describes the appearance on different platforms, and logic defines dependencies between form elements and some additional functionalities. XForms enables the validation of user input on the client within the browser. This reduces client-server interactions and server load. Furthermore, XForms provides means to calculate values from user input on the client device. Most mobile devices, however, do not currently support special browsers for viewing XForms. In RIML, XForms is used to handle form's and data modeling. SMIL is used to enable the definition and selection of content depending on the device type.

The architecture of the Adaptation Engine (AE) that is used for converting RIML to appropriate formats is displayed in figure 6.21. It was developed with the help of Java EE and applies Xerces [ASF05d], Xalan [ASF05c], DELI [HP05], and Chiba XForms processor [Turn03]. The engine consists of an Adaptation Controller, an Adaptation Pipeline and some modules supporting session and context information [cf. Cons04, pp. 98-100; Zieg+04]. The Controller is in charge of request processing and forwards requests to appropriate components. It communicates with the components responsible for storing device and user information. It also interacts with the SessionContext Container to determine an appropriate adaptation for a particular device.

In the adaptation process, the Adaptation Pipeline is the most relevant component. It is composed of six elements: a Reducer, a Paginator, an XForm Processor, a Markup Mapper, a Stylist, and a Validator. The Reducer selects the content to be displayed basing its decision on the device characteristics. The Paginator is in charge of splitting content into smaller units displayable on one screen and the generation of navigation links. The generated pages are stored in the Pagination Store, from which all the pages are retrieved. The XForm Processor converts forms written in XForms language to a target markup, because mobile browsers do not support this standard. The Markup Mapper translates RIML into an appropriate end-format (HTML, XHTML, WML, or VoiceXML) by applying a suitable marker. The Stylist module applies style sheets to the generated markup language and the Validator validates the output before sending it to the client device.

RIML handles device-dependent content selection, pagination, layout, and navigation. The layout of RIML documents is based on three types of containers: rows, columns, and frames. The frame container can only contain content and does not include additional layout elements. Each layout definition consists of one or more frames. The columns and rows containers organize layout vertically and horizontally, respectively. These containers may also be nested. The containers can be paginating or non-paginating. If the container is non-paginating, it is displayed on one page; otherwise it is split into multiple fragments connected with navigation links. The pagination is accomplished according to an implemented pagination algorithm.

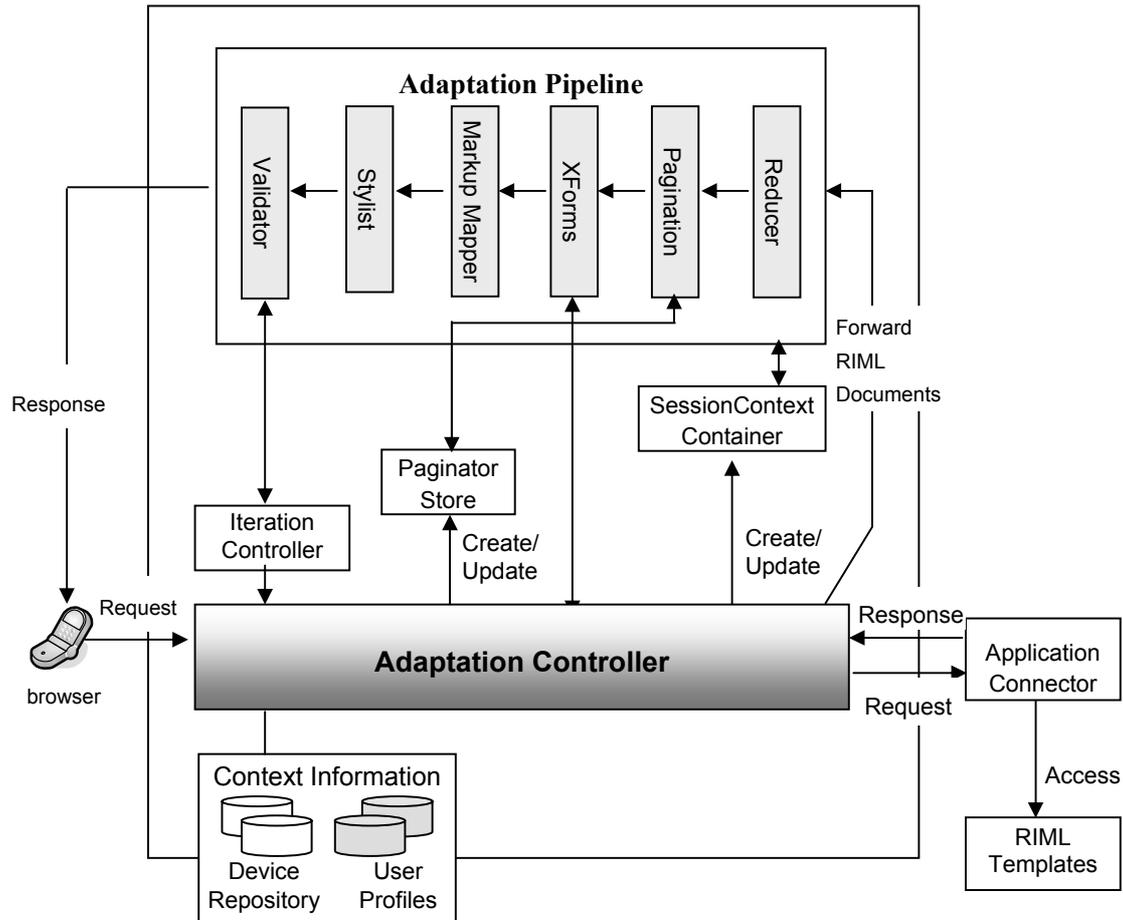


Figure 6.21: Architecture of RIML adaptation engine [own illustration]

The Adaptation Engine is an open-source software but it requires considerable installation and configuration effort in order to run properly [cf. Cons03]. Up to now, RIML lacks any available development environment that would facilitate the implementation process, although such support was announced. Without suitable development tools, it cannot be expected that RIML will gain widespread popularity and will be used by users without experience in Java EE programming.

6.3.3.3 Comparison of User Interface Description Languages

User Interface Description Languages may seem similar at the first sight, but they differ in many aspects. Table 6.2 provides a comparison of the aforementioned UIDLs in terms of supported models, methodology for UI description, available tools, supported languages, abstraction level, and context of use (user/environment/platform model). It furthermore specifies, whether they are open-source products or commercial developments.

The choice of a UIDL for a particular project depends on the goals pursued in this project and not only on the characteristics of the language. For example, XML and USXML can be regarded as the best

alternatives since both of them are open-source, meta-model languages supporting many models. However, they are not equipped with good tools and development environments. UIML renderers are pieces of commercial software, while RIML adaptation engine is provided at no cost¹⁰⁰. The main disadvantage of all languages is the fact that they cannot be used by an average user, who knows HTML, WML, or XHTML, to develop device-independent presentations because of the complicated language structure and a steep learning curve.

	Models	Methodology	Tools	Supported languages	Open-source	Target/ Abstraction level
UIML	Presentation and dialog models, partially domain model	Specification of multiple UI presentations, factoring/ corrections	Multiple rendering engines, code generator, editor	C++, Java, VoiceXML, HTML, WML, PalmOS, .NET	No (only language specification)	Multi-platform Model level
USIXML	Task, domain, user, dialog, presentation models	Specification of multiple UI descriptions or of generic description	Rendering engine, code editor	HTML, WML, Java	Yes	Multi-platform Meta-model level
DDL	Domain and presentation models	Specification of multiple UI presentations	Adaptation engine	WML, XHTML, HTML	No	Multi-platform Model level
RIML	Domain and presentation models	Specification of multiple UI presentations	Adaptation engine	WML, XHTML, HTML, VoiceXML	Yes	Multi-platform Model level
MyXML	Presentation and domain models	Specification of multiple UI presentations	Editor (developed but not available)	HTML, WML, XML	Yes	Multi-platform Model level
XIML	Task, domain, user, dialog, presentation models	Specification of multiple UI descriptions or generic description of UI	Rendering engine, code editor	HTML, WML, Java	Yes	Multi-platform, context-sensitive applications Meta-model-level

Table 6.2: Comparison of UIDLs [Berg+04; SoVa03]

¹⁰⁰ Cf. <http://sourceforge.net/projects/consensus>.

6.3.4 Frameworks supporting Model-View-Controller design pattern

A design pattern is “an idea that has been useful in one practical context and will probably be useful in others” [Fowl03]. Design patterns were known and appreciated by the software community, but they have become a common discussion topic after the publication of the book “Design Patterns: Elements of Reusable Object-Oriented Software” written by the so-called “Gang of Four” (GoF) - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides [Gamm+95]. The authors did not invent patterns; they carefully selected and described those they knew to be useful. The Model-View-Controller (MVC) design pattern is one of the most popular patterns introduced in the 80s and is also applied in Java EE architectures. This section introduces this pattern and describes some relevant device-independent approaches based on it.

6.3.4.1 Model-View-Controller design pattern

The Model-View-Controller design pattern, a triumvirate introduced in the 1980s in the ObjectWorks/Smalltalk programming environment, contains three separate objects: a Model, a View, and a Controller (cf. figure 6.22). The Model is in charge of maintaining the data and the state of an application and contains the core application’s functionality. It is completely separated from the View and Controller and does not know anything about them. The View is a user interface that presents information about the Model to the user. It does not possess any knowledge about the Controller.

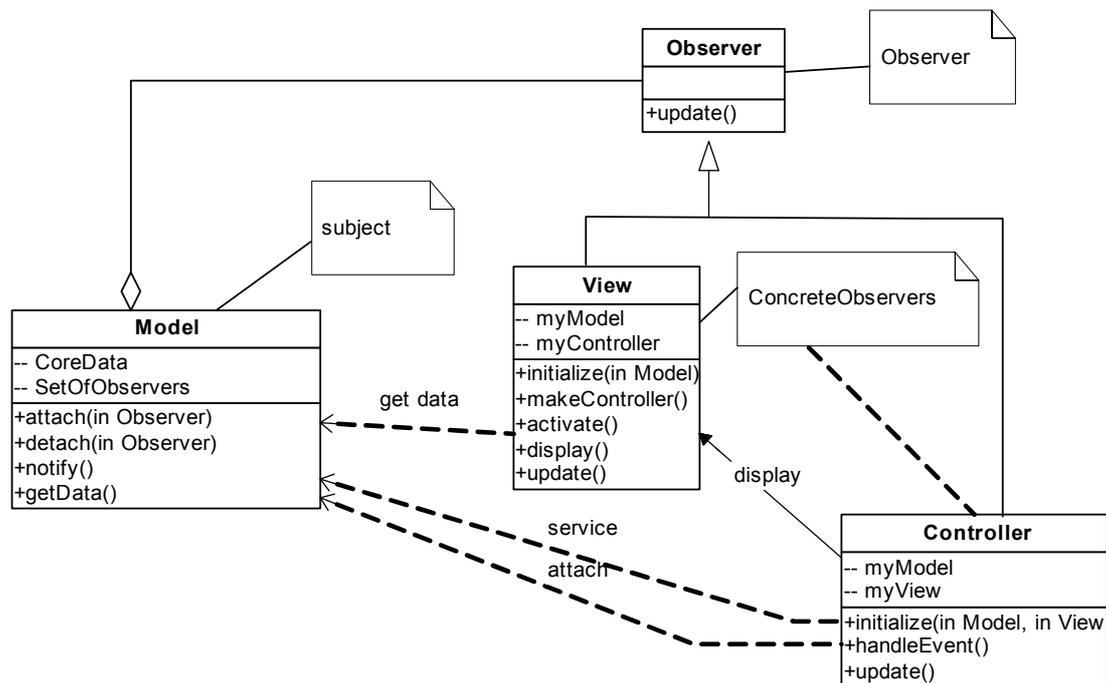


Figure 6.22: MVC design pattern [ShTr01, p. 343].

When important changes take place in the Model, all of its views are updated. The Controller interacts with the user – it accepts user input, influences the Model, and causes the View to be changed in a suitable manner. The Controller works as an interaction layer between the presentation layer and the data layer. Multiple views can therefore be attached to the application, without rewriting its code. In

architectures based on the Java EE technology, Views are in the form of JSP pages, servlets play the role of Controllers and JavaBeans represent Models (the so-called Java EE Model 2 architecture) [John+02].

6.3.4.2 Exemplary MVC approaches

Device-independent applications are built upon the same data and business logic and differ only with regard to presentation layers (views). Many approaches for server-side content adaptation are therefore based on the Model-View-Controller (MVC) design pattern. This section presents some relevant implementations of this pattern, for example Struts [cf. ASF05; Cava04; Hust+03], Java Server Faces [cf. DuLeWi03; Sun04c] or Multi-Device Authoring Technology (MDAT) [cf. Bana+04; Bana04a].

Struts

Struts is an open-source framework consisting of servlets, cooperating utility classes (for tasks such as internationalization or XML parsing), and JSP tags [Cava04; Hust+03]. It is based on the MVC Model 2 architecture and, additionally, on different design patterns such as Service to Worker, Front Controller, Singleton, Dispatcher, View Helper, Value Object, Composite View, and Synchronizer Token [cf. Hust+03, pp. 577-581].

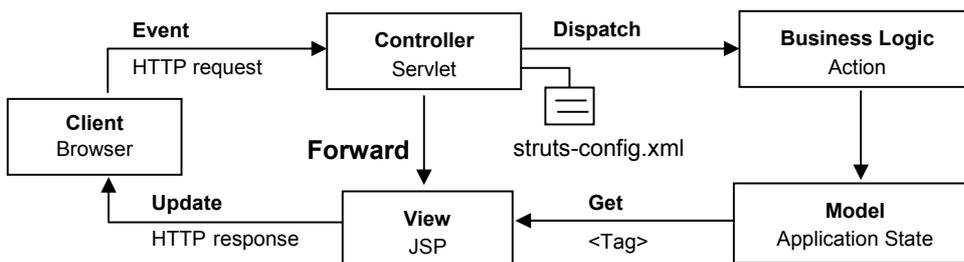


Figure 6.23: Struts overview

Figure 6.23 displays the main components of the framework that works in the following way: A client browser sends an HTTP request and creates an event to which a Web container (e.g. Apache Tomcat [ASF05b]) responds. The Controller, implemented as a servlet, receives the request and dispatches it according to the information stored in the configuration file (“struts-config.xml”). An Action class updates the state of the Model and is responsible for the flow of the application. It can validate user inputs and possesses access to the business layer to retrieve information from databases and other data services. The Model updates the application state. The Model state (at the request or session level) is represented by an ActionForm bean from which information is read. The View can use a set of implemented tags and is realized as a JSP page.

The developed tag libraries include:

- the struts-html tag library for creating dynamic HTML user interfaces and forms
- the struts-bean tag library for providing additional functionality to the `<jsp:useBean>` tag
- the struts-logic tag library for implementing conditional statements
- the struts-template tag library for applying tags that are useful for the creation of dynamic JSP templates for pages that share a common format.

Struts facilitates the development of device-independent applications because it separates the presentation layer from the business and data layers. Unfortunately, open-source tag libraries that would facilitate the development of mobile applications are not available so far and the applications have to be developed from scratch. The framework divides the work between programmers and designers and consists of reusable components that can be used in various programs.

JavaServer Faces (JSF)

JavaServer Faces (JSF) is an application framework for creating Web-based user interfaces [cf. Berg04; DuLeWi03; Sun04c]. It is currently part of the Java 2 Enterprise Edition and provides an API for generating UI components and managing their states. The framework helps to handle events and server-side validation, supports internationalization, data conversion, and the definition of page navigation. The UI components can be created with the help of JSP custom tag libraries. JSF technology supports the MVC Model 2, but its main focus is on UI components and events.

Figure 6.24 presents a Unified Modeling Language (UML) class diagram for the main components of JavaServer Faces and their relationships [cf. Mann05, pp. 39-57]. UI components are contained in a view update beans that generate events based on user inputs. Renderers render UI components and are able to generate events and messages. Converters translate values for components and generate error messages. Validators are in charge of validation of components' values. In backing beans event listeners and action methods (event listeners specialized for navigation) are placed. Event listeners manipulate the view or execute model objects containing business logic. Action methods return some output that is used by a navigation system to select the appropriate view.

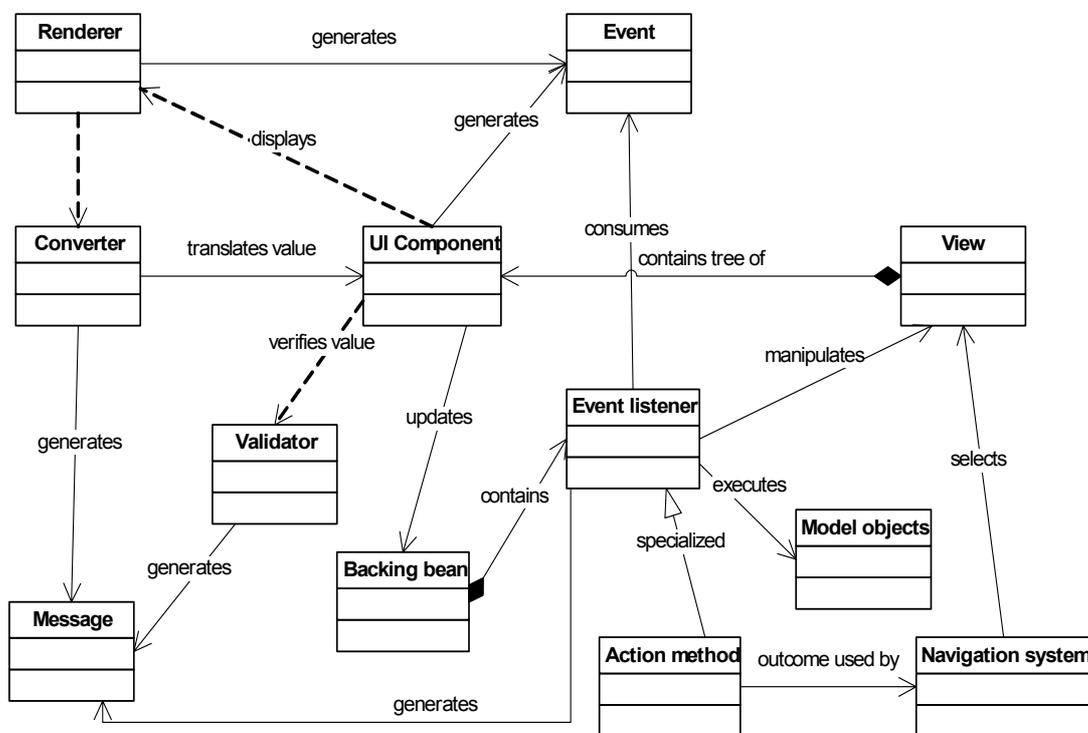


Figure 6.24: UML class diagram representing JSF components [Mann05, p. 40]

JavaServer Faces were developed to produce markup for HTML pages, but can be extended to generate additional markups such as WML or to create Java ME applications. For example, in the book “Core JavaServer Faces” Geary and Horstmann show how to develop a tag library for displaying Java ME UI components [GeHo04, pp. 505-560]. Furthermore, within the scope of the MyFaces project [ASF105], a tag library for rendering WML components named WAP tags and a tag library for enhancing standard JSP actions and custom MyFaces actions (with the “x” prefix) were provided¹⁰¹. The JavaServer Faces framework has advantages similar to Struts. Additionally, it provides a rich architecture for managing the state of components, processing and validating data, and handling events.

Multi-Device Authoring Technology (MDAT)

Multi-Device Authoring Technology (MDAT) is a development methodology and a toolset based on the MVC design pattern and supporting the development of form-based Web applications for heterogeneous devices [Bana+04; Bana+04a]. It is a successor of previous research prototypes, mainly of the Platform-Independent Model for Applications (PIMA) [cf. Bana+00; GaBeLa03].

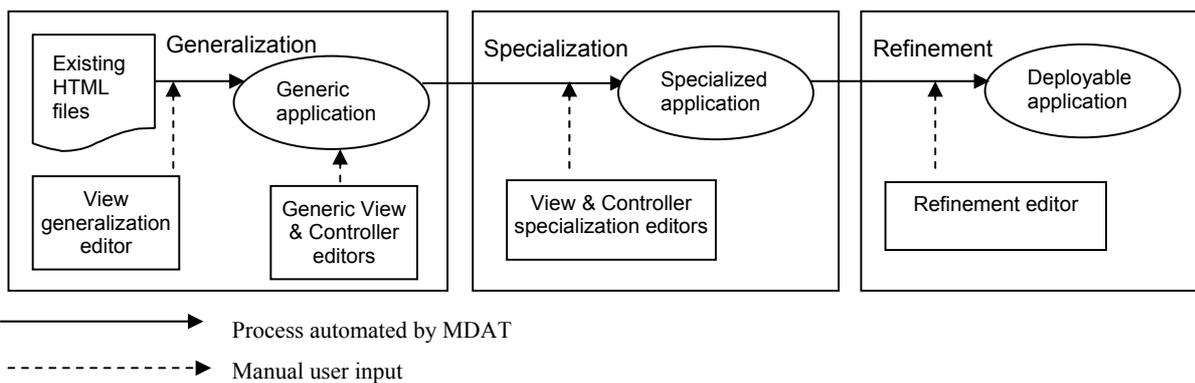


Figure 6.25: Design-time application generation in MDAT [Bana+04, p. 86]

In MDAT, a developer defines a generic application representing common aspects of the View and Controller components across various devices. Generic applications can be created manually (with generic View and Controller editors) or automatically from existing HTML pages. Generic applications are translated into device-specific applications. MDAT transforms the Controller into a set of Struts action classes and forms beans. It also produces a set of JSP pages for multiple devices. In the last step, MDAT creates standardized, deployable Web applications. The developer can change the presentation for specific devices (e.g. page formatting, layout) in the Refinement Editor. The whole process is depicted in figure 6.25. Device profiles and categories can be generated and edited with the help of device profile tools. MDAT was released as a component of the Everyplace Toolkit for WebSphere Studio¹⁰². A usage study showed that the framework is quite complex and can be

¹⁰¹ Cf. <http://incubator.apache.org/myfaces/tlddoc/> for a detailed description of tags.

¹⁰² Cf. http://www.ibm.com/software/pervasive/everyplace_toolkit.

successfully used only by developers with previous experience with the MVC design pattern [Ban+04, pp. 92-93].

Multichannel Object REnderer (MORE)

Multichannel Object REnderer (MORE) is a framework for automatic generation of user interfaces from a reflective analysis of data and Java code that represent the application model [Carb+02; Carb+02a; Carb+03]. It can be applied to three categories of devices: fat clients (PCs, notebooks), thin clients (PDAs, smartphones), and Web-like clients (WAP-enabled phones).

MORE is based on the MVC design pattern. Data is traditionally bound to the Model component and a View-Controller subsystem is responsible for user interfaces. Since device-independent models are possible but device-independent views cannot be defined, the authors developed a framework for specifying and composing models from which appropriate views are generated. They assumed that each data type is a model of some kind of object. For example, a String value is a model of a text object and can be rendered as a text box. MORE includes a set of classes for platform-independent UI models consisting of “basic models” and “composite models”. Basic models do not have to be further decomposed to generate a user interface. Composite models are rendered recursively until atomic components are obtained. With the help of the Java reflection mechanism the model is inspected at the runtime and, depending on the device, a concrete view is built¹⁰³.

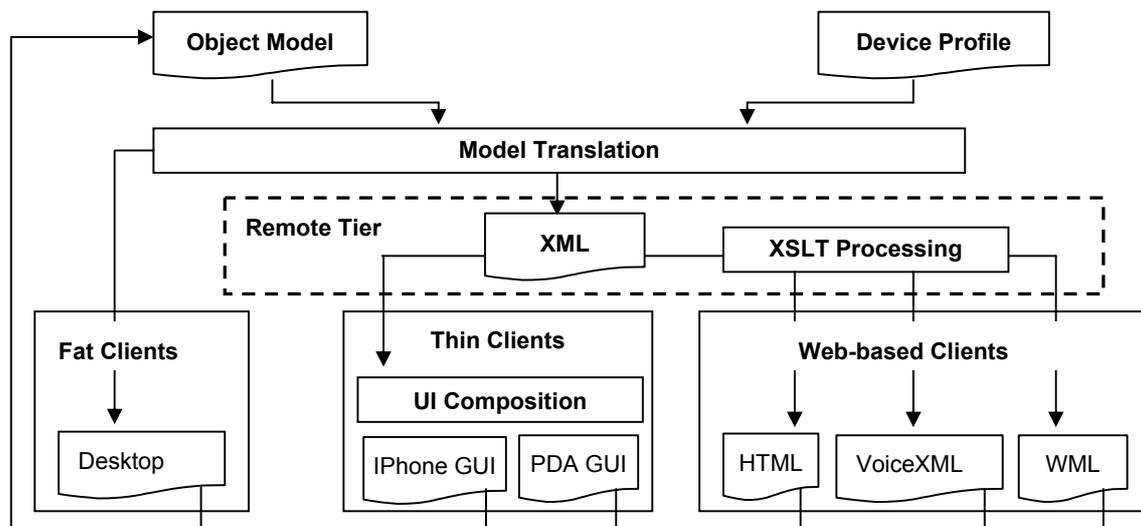


Figure 6.26: MORE architecture [Carb+03, p. 188]

MORE generates user interfaces for different clients in heterogeneous ways. For fat clients, model introspection and the construction of interfaces (e.g. in Swing) is performed on the client. The engine analyzes the properties and methods of the model and renders a suitable GUI representation (Java

¹⁰³ Reflection is a feature of Java programming language that allows programmatic access to loaded Java classes (without previous knowledge about the classes). It is, for example, possible to retrieve information about modifiers, fields, methods, constructors, and superclasses of a class. Additionally, the loaded classes can also be manipulated (e.g. values of fields may be changed, etc.) [cf. Ecke03, pp. 449-478].

ME, PersonalJava, etc.). For thin and Web-based clients, an XML representation of the model is created. A thin client parses this XML and generates a user interface from it. For Web-based clients an XSLT processor produces appropriate markup languages from XML (e.g. HTML, WML, or VoiceXML).

MORE was used for the development of multi-channel applications in the E-MATE project [Carb+02a] and was evaluated by developers engaged in this project. Although the framework is responsible for the generation of device-independent user interfaces and is easy to extend, it lacks fine customization of these user interfaces and full control over the presentation.

6.4 Comparison of approaches

This section provides a comparison of some chosen approaches for device-independent content delivery from all of the categories presented in the previous sections (client-side, server-side, and intermediary adaptation). Nine approaches were evaluated using the Device Independence Principles and the Device Independence Authoring Challenges: m-Links, Opera Mini, CSS/SSR, Cocoon, UIML, RIML, DDL, myXML, and MDAT.

Approach	DIP-1: Device - independent access	DIP-2: Device-independent Web page identifiers	DIP-3: Functionality	DIP-4: Incompatible access mechanism	DIP-5: Harmonization
Device Independence Principles (DIP)					
m-Links	**	*	**	***	***
Opera Mini	**	***	**	**	**
CSS/SSR	**	***	**	*	**
Cocoon	***	**	***	**	***
UIML	***	***	***	**	***
RIML	***	***	***	**	***
DDL	***	***	***	**	***
MyXML	***	***	**	*	***
MDAT	***	***	***	**	***
Evaluation scores: *: no support **: functionality only partially supported ***: functionality supported					

Table 6.3: Evaluation of content adaptation methods with regard to Device Independence Principles [own research]

With regard to the Device Independence Principles, client-side adaptation approaches came off badly and approaches that perform the adaptation on the server-side were evaluated as particularly good and suitable (cf. table 6.3). Client-side adaptation, for example, the Small-Screen Rendering based on the CSS technology, does not provide adequate mechanisms for handling incompatible accesses to the content and displaying appropriate error messages (DIP-4). Furthermore, it only offers limited

functional presentation. The customization of content is not always possible since the transformation process is performed only on devices with a CSS support and the resulting pages are quite long, with scaled images and fonts and with complex navigation structures.

In server-side adaptation approaches, the only principle that is not fully supported is the incompatible access mechanism. The developers themselves should take care of error messages if the user tries to access the content with devices for which the adaptation is not possible. The implementation of such messages is principally achievable, but it is not provided automatically by these frameworks.

With regard to a selected set of the Device Independence Authoring Challenges, client-side adaptation performed worst and server-side adaptation, particularly RIML and MDAT, gained the highest scores. The criteria were grouped into six categories (cf. table 6.4 and 6.5):

- 1) criteria with regard to the comprehensibility of scope (e.g. application scope, integration of device-dependent and device-independent content)
- 2) factors with the focus on frameworks' extensibility
- 3) criteria that take into account the simplicity of approaches and content
- 4) criteria supporting context variability (e.g. navigation or media variability)
- 5) criteria which account for the variability specified by authors (e.g. different layouts)
- 6) criteria with the focus on affordability (cost, effort required to learn and apply a framework)

Server-side adaptation approaches, especially RIML, DDL, MyXML, and MDAT, offer a very good support for the variability of delivery context and author-specific variability. They support the generation of different navigation mechanisms, enable the presentation of various media types and content depending on the device type. In these approaches, the layout can vary and presentation units can be aggregated or defragmented (split into smaller units), depending on the delivery context. The frameworks are extensible and it is possible to scale the complexity of applications. However, the application scope is relatively small but the complexity of such frameworks and the effort required to learn them is very high. The scalability of effort is almost not possible because the authors cannot influence the time and resources invested in the application development and are not able to minimize the efforts.

The affordability of the frameworks is evaluated as "middle" because they are usually developed as open-source projects. However, most of them were implemented as prototypes and cannot be found any longer on the net (for example, the implementation of RIML disappeared at the end of 2005). Paradoxically, two Opera's technologies (Opera Mini and SSR) that offer only a limited support for the variability of delivery context and author-specific variability, but are simple, scalable, and easily affordable seem to be more successful and popular than the server-side approaches with their great functionality.

Challenge/ Evaluation criteria	Approach	m-Links	Opera Mini	CSS/SSR	Cocoon	UIML
Provide comprehensive scope						
DIAC-3.1: Application scope		*	***	-	**	***
DIAC-3.19;3.20: Integration of device-dependent and independent content		*	-	-	***	*
DIAC-3.28: Range of complexity		***	**	**	***	****
Support smooth extensibility						
DIAC-3.2: Extensible capabilities		***	-	*	***	***
DIAC-3.29: Scalability of complexity		**	-	**	***	***
Support simplicity						
DIAC-3.4: Simplicity		****	****	***	**	**
DIAC-3.11: Simple content		****	***	-	***	***
Support delivery context variability						
DIAC-3.5: Navigation variability		***	**	*	***	***
DIAC-3.6: Organization variability		***	**	*	***	***
DIAC-3.7: Media variability		***	**	**	***	***
Support variability specified by authors						
DIAC-3.12: Text content variety		***	*	*	***	**
DIAC-3.13: Media resource variety		***	*	*	**	***
DIAC-3.14;3.15: Media resource specification & selection		**	*	**	**	***
DIAC-3.30;3.31: Aggregation & decomposition		***	**	-	**	***
DIAC-4.3: Layout variety		***	*	**	**	***
Affordability						
DIAC-3.3: Affordability (cost)		***	****	****	***	**
DIAC-3.33: Reusing existing applications		***	****	***	***	***
DIAC-6.5: Minimization of effort		**	****	***	**	**
DIAC-6.13: Separation of device-dependent and device-independent material		**	*	*	**	**
DIAC-6.6: Abstraction of device knowledge		*	****	-	*	**
DIAC-6.10: Scalability of effort/quality		**	***	***	***	***
Scale: *: non existing; **: low; ***: middle; ****: high; -: does not apply to this method						

Table 6.4: Comparison of content adaptation methods with the help of Device Independence Authoring Challenges (DIACs) [own research]

Approach	RIML	DDL	MyXML	MDAT
Challenge/ Evaluation criteria				
Provide comprehensive scope				
DIAC-3.1: Application scope	***	***	**	***
DIAC-3.19;3.20: Integration of device dependent and independent content	**	**	***	***
DIAC-3.28: Range of complexity	****	****	****	****
Support smooth extensibility				
DIAC-3.2: Extensible capabilities	***	***	***	***
DIAC-3.29: Scalability of complexity	***	***	***	***
Support simplicity				
DIAC-3.4: Simplicity	**	**	**	**
DIAC-3.11: Simple content	***	***	***	***
Support delivery context variability				
DIAC-3.5: Navigation variability	****	****	****	****
DIAC-3.6: Organization variability	****	****	****	****
DIAC-3.7: Media variability	****	****	****	****
Support author specified variability				
DIAC-3.12: Text content variety	**	**	**	**
DIAC-3.13: Media resource variety	****	***	***	***
DIAC-3.14;3.15: Media resource specification & selection	****	***	***	***
DIAC-3.30;3.31: Aggregation & decomposition	****	****	****	****
DIAC-4.3: Layout variety	****	***	***	***
Affordability				
DIAC-3.3: Affordability (cost)	***	***	***	**
DIAC-3.33: Reusing existing applications	***	***	***	***
DIAC-6.5: Minimization of effort	**	***	***	***
DIAC-6.13: Separation of device-dependent and device-independent material	***	***	***	***
DIAC-6.6: Abstraction of device knowledge	****	***	***	***
DIAC-6.10: Scalability of effort/quality	**	**	**	**
Scale: * : non existing; ** : low; *** : middle; **** : high; - : does not apply to this method				

Table 6.5: Comparison of content adaptation methods with the help of Device Independence Authoring Challenges (DIACs) [own research]

7

Common elements of frameworks

The transformation of images and the identification of the delivery context are two common features that should be supported by each adaptation framework. In this research work, two open-source libraries were used in the prototypical implementations of frameworks to perform these tasks: a special Sun's API for the conversion of images (JIMI) and a library developed by Mark Butler for capturing device properties from CC/PP and UAProf (DELI). The delivery context is particularly important for adaptation approaches because it allows to determine the features of devices and helps to tailor content to these characteristics. It should be, however, mentioned that the number of handsets is continuously rising from day to day and there exists no database that would be able to deliver device properties for each device available on the market. Therefore, device-independent content adaptation is limited to the devices that can be found in registries of handsets and this situation will not change until handset makers will become more interested in the delivery of device context.

This chapter focuses on the possibilities to present and capture information about device features. Additionally, it presents some libraries that enable the transformation of images. In section 7.1 different approaches for delivery context are described and compared. The focal point of this section is the Composite Capabilities/Preference Profile (CC/PP) standard and the Delivery Context Library for CC/PP and UAProf (DELI). CC/PP is presently the best and most comprehensive approach for describing the context delivery. Section 7.2 gives a short overview of image formats and available open-source libraries for input/output (I/O) operations. It also outlines the most important features of the library chosen for the implementation of image transformation - JIMI.

7.1 Delivery context

The term delivery context is defined as "a set of attributes that characterizes the capabilities of the access mechanism, the preferences of the user and other aspects of the context into which a Web page is to be delivered" [W3C05a]. Context refers to all circumstances in which a computing task takes place such as available computing resources, spatial or temporal information (e.g. location, time, and date), features of physical environment and the physical, social or emotional state of the user. The research on context-aware applications aims at the design and development of systems that are able to respond to context changes and deliver context-sensitive information [DeSaAb01; Dey01; Jank03]. In the adaptation approaches proposed by the author of this thesis, the captured delivery context is limited to the information on interaction (input/output modalities), user agents (e.g. supported language and image formats), and connection characteristics (e.g. bandwidth), since the development of context-sensitive mobile applications that would react to environmental changes or spatial information is behind the scope of this research.

Delivery context information can be sent as part of the request or may be delivered indirectly by providing a reference to the data stored on a separate machine (e.g. on a server). Currently, there are several commonly known standards that help to obtain the delivery context and some further specifications that are being developed. Popular approaches include HTTP headers, W3C Composite Capabilities/Preference Profile (CC/PP), WAP User Agent Profile (UAProf), the SyncML Device Information standard (DevInf), Media Queries, the Universal Plug and Play standard (UPnP), Wireless Universal Resource File (WURFL), and Universal Profiling Schema (UPS) [cf. W3C02b; W3C05a]. There exist some Application Programming Interfaces (APIs) for processing contextual information. Delivery Context Library for CC/PP and UAProf (DELI), Wireless Abstraction Library (WALL), and Semantic API for the Delivery Context (SADiC) are the most commonly used APIs for dealing with the contextual information. For the delivery of mobile context, HTTP headers and CC/PP are usually applied [SpGo02; W3C02]. Interestingly, CC/PP standard enjoys increasing popularity among developers, but device manufacturers are still reluctant to implement it.

HTTP headers

Delivery context is usually extracted from the HTTP standard `Accept` headers [W3C05a]. These headers are as follows:

- `Accept` – contains information about the MIME types accepted by the User Agent
- `Accept-Charset` – indicates the preferred character set
- `Accept-Encoding` – indicates the encoding for the User-Agent
- `Accept-Language` – specifies the language set

The sample HTTP request in listing 7.1 presents an example of `Accept` headers:

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
       application/vnd.ms-powerpoint, application/vnd.ms-excel,
       application/msword, */*
Accept-Language: en-ca
Accept-Encoding: gzip, deflate
```

Listing 7.1: Example of Accept headers

Many browsers do not provide correct information or the data is not complete. For example, the `Accept` header `*/*` asserts that the browser is able to accept any media type, although this is usually not true.

The `User Agent` header comprises data about device manufacturers, device version numbers, device hardware, and used browsers. Crucial contextual information included in the `User Agent` header is not standardized and depends on the good will of device manufacturers. An application can examine the `User Agent` header and perform adaptation, but such adaptation will be exclusively based on device/browser identity (e.g. Motorola A760, Microsoft IE 6.0, etc.) and not on particular capabilities of the device.

HTTP headers is a non-extensible format and can deliver ambiguous information. For example, Microsoft Internet Explorer identifies itself as Mozilla and Opera browser may send in the User Agent header the information that it is Opera, Microsoft Internet Explorer, or Mozilla browser, depending on the configuration provided by the user. HTTP also includes a mechanism for content negotiation. In the headers, the client can specify which media formats are supported and can indicate its preferences by providing the preference factor “q”. q can have a value between 0 and 1, where 1 indicates the highest priority and 0 the lowest one. Table 7.1 lists some sample HTTP headers and table 7.2 introduces the most common MIME types for mobile applications.

HTTP header example	Meaning
Accept-Charset: utf-8; q=0.9, iso-8859-1; q=0.5	UTF-8 character set is preferred, ISO-8859-1 can also be sent.
Accept: audio/*; q=0.2, audio/midi	Accept header with the following meaning: the user prefers audio/midi format, but any audio type can be sent if midi is not available.
Nokia6230/2.0(03.06) Profile/MIDP-2.9 Configuration/CLDC-1.1	User-Agent header for the Nokia 6230 built-in browser.
Mozilla/2.0 (compatible; MSIE 3.02; Windows CE; 240x320; PPC)	User-Agent header for Microsoft Internet Explorer installed on Pocket PC.

Table 7.1: Sample HTTP headers

File Type	MIME Media Type
Audio 3GPP files (.3gp)	audio/3gpp
Audio AMR files (.amr)	audio/amr
Audio AMR (wideband) files (.awb)	audio/amr-wb
Audio MIDI files (.mid or .midi)	audio/midi
Audio MP3 files (.mp3)	audio/mpeg
Audio MP4 files (.mp4)	audio/mp4
Audio WAV files (.wav)	audio/wav audio/x-wav
HTML files (.html or .htm)	text/html
Image BMP files (.bmp)	image/bmp image/x-bmp
Image GIF files (.gif)	image/gif
Image JPEG files (.jpg or .jpeg)	image/jpeg
Image PNG files (.png)	image/png
Image TIFF files (.tif or .tiff)	image/tiff

File Type	MIME Media Type
Image WBMP (Wireless BMP) files (.wbmp)	image/vnd.wap.wbmp
Java application JAR files (.jar)	application/java application/java-archive application/x-java-archive
Java application JAD files (.jad)	text/vnd.sun.j2me.app-descriptor
Plain text files (.txt)	text/plain
Symbian application SIS files (.sis)	application/vnd.symbian.install
Video 3GPP files (.3gp)	video/3gpp
Video MP4 files (.mp4)	video/mp4
WML files (compiled) (.wmlc)	application/vnd.wap.wmlc
WML files (plain text) (.wml)	text/vnd.wap.wml
WMLScript files (compiled) (.wmlsc)	application/vnd.wap.wmlscriptc
WMLScript files (plain text) (.wmls)	text/vnd.wap.wmlscript
XHTML MP files (.xhtml, .html or .htm)	application/vnd.wap.xhtml+xml application/xhtml+xml text/html

Table 7.2: Common MIME types [<http://www.developershome.com/wap/detection/detection.asp?page=httpHeaders>]

In order to retrieve all available HTTP headers using Java, the code shown in listing 7.2 can be applied:

```

1. try {
    // Create a URLConnection object for a URL
2.     URL url = new URL("http://www.someURL.com");
3.     URLConnection conn = url.openConnection();
    // List all the response headers from the server.
4.     for (int i=0; ; i++) {
5.         String headerName = conn.getHeaderFieldKey(i);
6.         String headerValue = conn.getHeaderField(i);
7.         if (headerName == null && headerValue == null) {
            // No more headers
8.             break; }
            // The header value contains the server's HTTP version
9.         if (headerName == null) { }
10.        System.out.println(headerName + " " + headerValue);
11.    }
12. } catch (Exception e) {
13. }

```

Listing 7.2: Retrieving HTTP headers in Java

Some interesting headers include the available screen size in pixels (the `X-UP-DEVCAP-SCREENPIXELS` attribute), the number of soft keys (the `X-UP-DEVCAP-NUMSOFTKEYS` attribute), and a URL to the UAProfile (the `X-WAP-PROFILE` attribute).

CC/PP

The Composite Capabilities/Preference Profile (CC/PP) standard [Butl02; W3C04a], recommended by the World Wide Web Consortium, delivers contextual information related to device specification and user preferences. The standard is based on the Resource Description Framework (RDF) [W3C04d], serialized to the XML format. CC/PP is vocabulary-independent: any vocabulary described by an RDF Schema [W3C04e] may be used. CC/PP is composable/decomposable and provides the possibility to dynamically create context profiles from fragments of capabilities information, distributed among multiple repositories on the Web. A CC/PP profile can describe a number of components (e.g. a mobile browser) by specifying their attributes (e.g. type, version and name of a browser, supported markup languages, a screen size, etc.).

```

1. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2.   xmlns:ccpp="http://www.wapforum.org/profiles/UAPROF/ccppschem-20010426#"
3.   xmlns:clt="http://marks.profile.org/2001/05-clt#">
4.   <rdf:Description about="http://www.profiles.org/jornada1000">
5.     <ccpp:component>
6.       <rdf:Description ID="BrowserUA">
7.         <rdf:type
8.           resource="http://www.wapforum.org/profiles/UAPROF/ccppschem-
9.             20010426#BrowserUA"/>
10.        <ccpp:CcppAccept>
11.          <rdf:Bag>
12.            <rdf:li rdf:parseType="Resource">
13.              <clt:specifier>text/html</clt:specifier>
14.              <clt:q>1.0</clt:q>
15.            </rdf:li>
16.            <rdf:li rdf:parseType="Resource">
17.              <clt:specifier>text/plain</clt:specifier>
18.              <clt:q>0.8</clt:q>
19.            </rdf:li>
20.            <rdf:li rdf:parseType="Resource">
21.              <clt:specifier>image/jpeg</clt:specifier>
22.              <clt:q>0.6</clt:q>
23.            </rdf:li>
24.          </rdf:Bag>
25.        </ccpp:CcppAccept>
26.      </rdf:Description>
27.    </ccpp:component>
28.    [...]
29.  </rdf:Description>
30. </rdf:RDF>

```

Listing 7.3: Fragment of CC/PP profile serialized to XML

A CC/PP profile consists of a number of components with various attributes. Listing 7.3 presents a simple CC/PP profile, serialized to XML. It is an equivalent of the following HTTP header: `Accept: text/html; q=1.0, text/plain; q=0.8, image/jpeg; q=0.6`. According to this delivery context, the browser supports the HTML format or eventually plain text and accepts JPEG images.

Efficient delivery of contextual information is achieved by using a specific mechanism. The client sends a reference to a profile stored on a separate computer serving as a profile repository. Additionally, it lists all properties that override the standard specifics of a particular client. In the process of profile resolution, profile references and differences are interpreted and the final profile is constructed. This saves bandwidth because only parts of the profile have to be sent with each request.

User Agent Profiles (UAProf)

The User Agent Profiles (UAProf) standard was defined by the WAP Forum as part of the WAP specification [WAPF01]. UAProf is an implementation of CC/PP for WAP-enabled mobile terminals and is transported with the help of CC/PP headers over HTTP. It has a two-level hierarchy, composed of components and attributes with specified vocabulary and can be written in RDF serialized to XML. Profiles that use the UAProf vocabulary consist of six components: HardwarePlatform, SoftwarePlatform, NetworkCharacteristics, BrowserUA, WapCharacteristics, and PushCharacteristics. HardwarePlatform describes hardware features such as screen size or color capability. SoftwarePlatform provides information about operating systems, MIME types, character sets, and audio/video encoders. NetworkCharacteristics gives an overview of supported network characteristics, BrowserUA describes the browsers (e.g. supported markup languages, information about the browser's vendor). WapCharacteristics characterizes the deck size and WAP/WML versions, PushCharacteristics reports about the allowed size of push messages and supported push content types. The most relevant attributes for HardwarePlatform, SoftwarePlatform, and BrowserUA are displayed in tables 7.3, 7.4 and 7.5.

The UAProf standard proposes a specific transfer syntax for profile and profile differences, as well as for resolution rules for default values of properties and difference values. First commercial implementations of UAProf are already available for WAP proxies and browsers. Moreover, the Mobile Execution Environment group within the European Telecommunications Standards Institute (ETSI) has agreed to adopt UAProf as a standard for device capabilities in future wireless devices [SaHj01, p. 42]. A repository with UAProf profiles can be found under http://w3development.de/rdf/uaprof_repository/.

UAProf Attribute / Property	Description
BitsPerPixel	Specifies the number of bits that one pixel uses to represent colors. This attribute can be used to calculate the number of colors supported. For example, if BitsPerPixel is 16, the mobile device can display $2^{16} = 65536$ colors.
ColorCapable	States whether the screen of the mobile device can display colors.
CPU	Specifies the name and model of the CPU used by the mobile device.
ImageCapable	States whether the mobile device can display images.
InputCharSet	Determines the character sets that can be used for text input.
Keyboard	Specifies the keyboard type of the mobile device.

UAProf Attribute / Property	Description
Model	Gives the model of the mobile device.
NumberOfSoftKeys	Asserts the number of softkeys available on the mobile device.
OutputCharSet	States the character sets that can be used for text output.
PixelAspectRatio	Gives the ratio of the width of one pixel to the height of one pixel. For example, if the ratio is 1:1, you will find the value 1x1 in PixelAspectRatio.
ScreenSize	Gives the screen size in pixels (screen width is the first value, screen height the second one).
ScreenSizeChar	States the screen size in characters, the largest character of the standard font of the mobile device is used as the unit. The first value is the screen width and the second value is the screen height. For example, the ScreenSizeChar of 18x5 means that the screen can fill 18 characters in a row and 5 characters in a column.
SoundOutputCapable	States whether the mobile device can output sound.
StandardFontProportional	Provides the information whether the standard font used by the mobile device is proportional.
TextInputCapable	States whether a user can input text (and not only numbers, what is the default setting).
Vendor	Provides information about the manufacturer of the device.
VoiceInputCapable	States whether the mobile device supports voice input.

Table 7.3: UAProf attributes/properties - Hardware Platform [WAPF01]

UAProf Attribute / Property	Description
AcceptDownloadableSoftware	States whether the user wants to accept downloadable software.
AudioInputEncoder	Specifies the audio input encoders supported by the wireless device.
CcppAccept	Contains a list of acceptable MIME media types.
CcppAccept-Charset	Contains a list of supported character sets.
CcppAccept-Encoding	States the transfer encodings acceptable by the wireless device.
CcppAccept-Language	Specifies the language preference of the user. The items are listed in the order of preference.
DownloadableSoftwareSupport	Specifies the downloadable software types that the wireless device accepts and executes.
JavaEnabled	Specifies whether the wireless device is capable of running Java programs.
JavaPlatform	Specifies Java technologies that are supported by the wireless device.
JVMVersion	States the name and version of the Java virtual machines.
OSName	Provides information on the name of the operating system of the wireless device.
OSVendor	States the vendor of the operating system of the wireless device.
OSVersion	States the version number of the operating system of the wireless device.
SoftwareNumber	Specifies the version number of the firmware.
VideoInputEncoder	States the supported video input encoders.

Table 7.4: UAProf attributes/properties - Software Platform [WAPF01]

UAProf Attribute / Property	Description
BrowserName	States the name of the browser that sent the current request.
BrowserVersion	Specifies the version number of the browser that sent the request.
DownloadableBrowserApps	States the application types that are supported by the browser of the mobile device.
FramesCapable	States whether frames are supported by the browser.
HtmlVersion	Gives the HTML version supported by the browser.
JavaAppletEnabled	Provides information on the support of Java Applets by the device.
JavaScriptEnabled	Specifies whether JavaScript is supported by the browser.
JavaScriptVersion	Specifies the supported JavaScript version.
PreferenceForFrames	Asserts whether the user prefers to use frames in HTML/XHTML pages.
TablesCapable	States whether tables are supported by the browser.
XhtmlModules	Lists the supported XHTML modules.
XhtmlVersion	Specifies the XHTML version supported by the browser.

Table 7.5: UAProf attributes/properties – Browser User Agent [WAPF01]

WURFL

Wireless Universal Resource File (WURFL) [Pass03] is an open-source project¹⁰⁴. It collects information about wireless devices and maintains XML configuration files with data about the capabilities of devices stored in them. Over 5000 variants of devices are currently saved in WURFL but the focus is put on European devices. The maintained profiles contain also additional device features that cannot be found in the UAProf specification. WURFL supports the inheritance of properties from other devices. Profiles for new phones can therefore be constructed by extending existing profiles. This is guaranteed by the fall_back mechanism: In a generic model, all properties are saved only with their minimal values. Devices can furthermore be categorized according to families and subfamilies. By default, new devices inherit the characteristics of the class to which they belong and its specific properties can be overridden. In the worst case, a generic profile will be found if no other data is available. Since WURFL is an open-source project, everybody can add new profiles to the repository or correct the available information.

Listing 7.4 shows an example of a WURFL profile for different versions of the Nokia 7110 handset. The generic model for Nokia 7110 does not support tables. There is, however, a family of devices that supports tables.

WURFL devices contain the `user_agent` string, the `fall_back` attribute, and a unique identification string (`id`). WURFL supports over 300 capabilities. Capabilities are collected in groups for better readability. The following groups are known: `product_info`, `wml_ui`, `chtml_ui`, `xhtml_ui`, `markup`, `cache`, `display`, `image_format`, `bugs`, `wta`, `security`, `storage`, `object_download`, `drm`, `streaming`, `wap_push`, `mms`, `sms`, `j2me`, and `sound_format` [cf.

¹⁰⁴ Cf. <http://wurfl.sourceforge.net>.

Tras05 for more details]. They describe the most important capabilities of devices such as the characteristics of user interfaces for different browsers (`wml_ui`, `xhtml_ui`, etc.), display features, support for Java ME or MMS, supported markup languages, image formats, etc.

To query the repository of WURFL XML files, a WURFL Java API was developed. The API has two main functions: it can return a WURFL device ID for a given User Agent string and a capability value for a given capability name and WURFL device ID. WURFL APIs for Perl, MS. Net, Ruby, and Python were also provided.

```

1. <device user_agent="Nokia7110/1.0 (04" fall_back="nokia_generic"
2.   id="nokia_7110_ver1">
3.   [...]
4.   <group id="ui">
5.     [...]
6.     <capability name="table_support" value="false" />
7.   </group>
8. </device>
9. <device user_agent="Nokia7110/1.0 (05.00)" fall_back="nokia_7110_ver1"
10.   id="nokia_7110_ver2">
11. <group id="ui">
12.   <capability name="table_support" value="true" />
13. </group>
14. </device>

```

Listing 7.4: Fragment of WURFL profile for Nokia 7110

Universal Profiling Schema (UPS)

Universal Profiling Schema (UPS) [cf. LeLa02; LeLa03; LeLa04] is a model that provides a comprehensive description of the delivery context. It includes six schemas: a client profile, a client resource profile, a document instance profile, a resource profile, an adaptation method profile, and a network profile. The client profile describes the general capabilities of a device, the client resource profile provides information about the device capabilities for a specific resource. The network profile introduces the network characteristics. The document instance profile describes the adapted document, the resource profile presents media resources, and the adaptation method profile describes the available adaptation method. The last three profiles refer to the server or proxy that is in charge of the adaptation process. UPS is based on CC/PP with an extended set of vocabularies.

The profiles repository¹⁰⁵ is deployed as a SOAP-based repository; the profiles can be retrieved using the Remote Procedure Calling (RPC) mechanism [Srin95]. With a Remote Procedure Call (RPC), one program can request a service from a program located on a different computer in a network without having to understand network details. Table 7.6 presents the Web Services methods that enable the retrieval and updating of profiles information in a repository, where 118 mobile devices are currently

¹⁰⁵ Cf. <http://wam.inrialpes.fr/people/lemlouma/MULIMEDIA/UPS-Client-Repository/ups-client-repository.html>.

stored. This is also the first implementation that uses Web Services to query UPS profiles and obtain delivery context.

Method	Parameters	Meaning
getProfile	profileID	Request a profile from a repository
getContextAtomicValue	profileID CCPPComponentID contextEntity	Request a value from a profile
getSubContext	profileID XPathExpression	Request a subcontext from a profile
updateContextAtomicValue	profileID CCPPComponentID contextEntity	Update a value in a profile

Table 7.6: Web Services methods for querying and changing UPS profiles [LeLa04, p. 109]

SyncML Device Information standard

The SyncML initiative¹⁰⁶ strives to develop a common synchronization protocol for exchanging data between servers and devices such as mobile phones, PDAs, and desktop PCs. Before the synchronization can take place, the devices have to exchange information about their capabilities. For this purpose, the SyncML Device Information standard (DevInf), implemented in XML, is used. The DevInf description delivers data about the device, its data storage and supported extensions as well as accepted content types.

Universal Plug and Play

Universal Plug and Play (UPnP)¹⁰⁷ is a standard supported by Microsoft. It is targeted at device-independent interconnections and uses XML to describe device capabilities. UPnP assumes that the devices will use XML together with XSL to manipulate the device description and will display, for example, only certain information about its capabilities.

Delivery Context Library for CC/PP and UAProf (DELI), JSR 188

DELivery Context Library for CC/PP and UAProf (DELI) is a Java library developed at HP Labs¹⁰⁸. It allows to resolve HTTP requests containing CC/PP or UAProf and to query the profiles. In DELI, the profile resolution is achieved by processing all referenced profiles in their order of presence, and by taking into account the profile differences. For each profile, or its differences, an RDF model is built and a list of attributes is extracted. From the produced vector of attributes, a new profile is constructed. DELI can be used in servlets and JSP pages. For context delivery handling, Sun also offers the

¹⁰⁶ Cf. <http://www.syncml.org>.

¹⁰⁷ Cf. <http://www.upnp.org>.

¹⁰⁸ Cf. <http://sourceforge.net/projects/delicon>.

Composite Capabilities/Preference Profile (CC/PP) Processing Specification API [Sun03; Sun04d] with a functionality similar to DELI.

In order to use the DELI library in a JSP page or a Servlet (cf. listing 7.5), it is necessary to include the DELI package and a `ServletContext` as a class data member (lines 1 and 3 in listing 7.5). When the servlet is initialized, the `ServletContext` should be queried in order to determine the path to the servlet and the obtained information has to be used to create a workspace (lines 6, 7, 10 in listing 7.5, respectively). After the initialization of the workspace, profile resolution can be performed by creating a new profile using a `HttpServletRequest` object (line 13). The profile can be manipulated by using the standard Vector methods which will retrieve profile attributes (cf. lines 14-16) or the profiles may be queried by using methods such as `get()`, `getAttribute()`, `getCollectionType()`, `getComponent()`, `getDefaultValue()`, `getResolution()`, `getType()` and `getValue()` [Butl02a].

```

1. import com.hp.hpl.deli.*;
2. public void init(ServletConfig config) throws ServletException {
3.     ServletContext servletContext;
4.     super.init(config);
5.     try {
6.         servletContext = config.getServletContext();
7.         Workspace.getInstance().configure(servletContext, "config/deliConfig.xml");
8.     } catch (Exception e) {}
9.     try {
10.        Workspace.getInstance().configure((ServletContext) null,
11.                                           "config/deliConfig.xml");
12.    } catch (Exception f) { } }

12. public void doGet(HttpServletRequest req, HttpServletResponse res)
13.     throws ServletException, IOException {
14.        Profile myProfile = new Profile(req);
15.        for (int i = 0; i < myProfile.size(); i++){
16.            ProfileAttribute p = (ProfileAttribute)myProfile.get(i);
17.            out.println("<TD>"+p.get()+"</TD>");}}

```

Listing 7.5: Retrieving CC/PP attributes with DELI [Butl02a]

Alternatively, the CC/PP profiles may be queried using a standard set of Java APIs for handling CC/PP information (JSR 188) [Mahm04; Sun03]. The JSR 188 reference implementation (JSR 188 RI) comprises three packages:

- `com.sun.ccpp` - provides the `DescriptionManager` class to configure the CC/PP Processing RI based on Jena Toolkit¹⁰⁹; it is used to manage CC/PP vocabulary description objects.

¹⁰⁹ Cf. <http://jena.sourceforge.net/>.

- `javax.ccpp` – contains classes and interfaces that enable the creation and access to CC/PP profile objects and represent CC/PP components and attributes. The entry point for most applications is the `ProfileFactory` class.
- `javax.ccpp.uaprof` - offers classes that represent UAProf attributes and values.

Semantic API for the Delivery Context (SADiC)

Semantic API for the Delivery Context (SADiC) [Cann03] is a Java API for interrogating and processing CC/PP and UAProf profiles with the help of semantic Web technologies. SADiC is able to perform the profile resolution process and does not rely on a specific set of vocabularies. In order to achieve the convergence between CC/PP and UAProf profiles, an ontology is used. Ontology is defined as a collection of definitions that describes concepts of a certain domain to enable the communication between machines¹¹⁰. In SADiC, the ontology is described in the Web Ontology Language (OWL) [W3C04c]. The library consists of a rich set of programming interfaces, offering access to the profiles data. Attribute values can be accessed directly as instances of specific data types. It is also possible to handle profiles that refer to multiple vocabularies or define own vocabularies/processing semantic.

7.2 Image converters

In the adaptation process, images have to be converted to formats supported by particular devices, scaled to fit on displays with various sizes, or reduced in order to improve the transfer and save the bandwidth. This section introduces the most important image formats and briefly describes some well-known Java libraries for image manipulation that can be used in the adaptation of images. Important features of the JIMI library that was chosen for the adaptation of images in the developed frameworks, are also discussed.

Image formats

Images can be divided into two distinct categories: bitmap files and vector graphics [cf. Mian99; Mury96]. Bitmap images are composed of a matrix of picture elements (pixels) with colors. True color, grayscale, black & white, and paletted can be distinguished as color types. In true color, a pixel can have three or more intensity values from color spaces such as RGB or CMYK¹¹¹. Grayscale images can store shades of gray from black to white. In bi-level (black&white) images, pixels can store two colors, black and white. Paletted (or color-mapped) images store one value per pixel. These values are taken from lists of colors, the so-called palettes.

¹¹⁰ Cf. <http://en.wikipedia.org/wiki/Ontology>.

¹¹¹ The RGB model stores values for red, green, and blue and describes what kind of light needs to be emitted to produce a given color. The CMYK model stores ink values for cyan, magenta, yellow, and black and describes what kind of inks need to be applied so the light reflected through the inks produces a given color [cf. http://en.wikipedia.org/wiki/Color_space].

The most popular bitmap formats are:

- Windows Bitmap (BMP) – a format commonly used by Microsoft Windows programs, a lossless compression can be specified but some programs use only uncompressed files
- Graphics Interchange Format (GIF) – a format used extensively on the Web, supports animated images
- Joint Photographic Experts Group (JPEG) – a format used mainly for internet graphics and photos on the Web
- Portable Network Graphics (PNG) - a 24 bit color-depth image format with a lossless compression designed to replace GIFs on the Web
- Tagged Image File Format (TIFF) - the most popular bitmap format for traditional print graphics with a lossy and lossless compression
- Wireless Application Protocol Bitmap (WBMP) [W3C99b] – a format used in WML on wireless devices.

Vector graphics are images that can be described with the help of mathematical definitions. Shapes in vector images are made up of a collection of points with lines interconnecting all of them, or a few control points that are linked by using so-called Bézier curves. Vector drawings are usually quite small files and can be scaled without quality loss. They can be stored among others in the following formats¹¹²:

- Encapsulated PostScript (EPS) – the most popular format for vector drawings
- Portable Document Format (PDF) - a versatile file format that can contain different types of data including complete pages
- Scalable Vector Graphics (SVG) – an XML-based language for describing two-dimensional graphics, it can generate static and dynamic vector graphic shapes, images, and text.

Open-source image libraries

There are plenty of open-source and commercial toolkits for the manipulation of images. Table 7.7 lists the most relevant open-source implementations and briefly characterizes each library. For the conversion and transformation of images, the Batik [ASF05], JIMI¹¹³, and JMagick¹¹⁴ libraries were taken into consideration.

JIMI and JMagick support a wide range of image formats. Batik is a Java library for viewing, generating, and manipulating SVG. It helps to convert SVG images into various graphic formats and to manipulate them (e.g. changing the size, resolution, and colors of images.). Since most mobile devices do not support SVG, this format can be converted to typical graphic formats that are displayable on mobile devices such as PNG or JPEG.

¹¹² Some file formats such as PDF or PS are also able to store vector and pixel images.

¹¹³ <http://java.sun.com/products/jimi/>.

¹¹⁴ <http://www.yeo.nu/jmagick/>.

Library / package, URL	License type	Description
Batik http://xml.apache.org/batik/	Apache Software License 1.1	Contains image I/O packages. It can read and write PNG, TIFF, and SVG files.
Image/J http://rsb.info.nih.gov/ij/	Free for non-commercial use	Reads and writes GIF, TIFF, and JPEG. Reads BMP, DICOM, FITS, and PGM.
ImageroReader http://reader.imagero.com/	Free for non-commercial use	Reads BMP, TIFF, PNG, JNG, MNG, JPEG, CRW, PSD, and GIF.
Java Advanced Imaging Image I/O API http://java.sun.com/products/java-media/jai/downloads/download-iiio.html	Sun license.	Reads and writes BMP, JPEG, JPEG 2000, PNG, PNM, Raw, TIFF, and WBMP image formats.
Java Image Loader http://www.vlc.com.au/~justin/java/images/	LGPL	Fast (using native code via JNI) and memory-efficient image reading for GIF, JPEG, PNG, TIFF, TGA, BMP, and XPM/XBM.
JIMI http://java.sun.com/products/jimi/	Sun Binary Code License Agreement	Sun development kit to read and write several image formats, including GIF, JPEG, TIFF, PNG, PICT, PSD, BMP, TGA, ICO, and CUR.
JIU - Java Imaging Utilities http://jiu.sourceforge.net/	GPL	Java imaging library with the support for PNG, GIF, IFF, RAS, PCD, PBM/PGM/PPM/PNM, PSD, TIFF
JMagick http://www.yeo.nu/jmagick/	LGPL	A wrapper to the functionality of ImageMagick, a powerful free imaging library supporting many formats.
Sanselan Java Image Library http://www.somethingremarkable.com/JavalImageLibrary.shtml	Apache	Reads PNG, GIF, TIFF, BMP, PSD, PBM/PGM/PPM.
WBMPMaster http://www.hut.fi/~jviinama/WBMPMaster.html	Free for evaluation	Writes WBMP files, can reduce images to black and white and edit them (scaling, flipping, mirroring).

Table 7.7: Java libraries for image manipulation [based on Schm05]

JMagick is an open source Java interface to ImageMagick - a free software for the creation and manipulation of bitmap images¹¹⁵. It is implemented as the Java Native Interface (JNI) into the ImageMagick API and therefore requires the installation of the ImageMagick software together with JMagick. JMagick can read, convert, and write images in a large variety of formats. It can resize, rotate, and change the colors of images, add supplementary elements like borders or frames to them, and describe the format and characteristics of images.

¹¹⁵ Cf. <http://www.imagemagick.org/>.

JIMI

The JIMI library was chosen for the adaptation of images in the developed frameworks because it offers support for various image formats, provides fast RGB random-access to pixel data, and possesses powerful features for handling images. JIMI can be used for image viewing, managing large or very-large images independently of a physical memory and can be extended to support new image types. Currently, the library supports GIF, JPEG, TIFF, PNG, PICT, Photoshop, BMP, Targa, ICO, CUR, Sunraster, XBM, and XPM formats.

The main classes in JIMI are responsible for mediating between user requests and managing images. JIMI contains a set of encoders and decoders. Figure 7.1 displays an overview of JIMI's most central classes. The `Jimi` class is a façade providing one-call access to the most universal image I/O operations. The `JimiReader` class is a lower level front-end for the communication with `JimiDecoders` and loading images. The `JimiWriter` class is responsible for the communication with `JimiEncoders` and image saving. The `JimiControl` class manages the pool of encoders/decoders. It is used for registering new format modules (decoders, encoders) and is able to find appropriate format modules based on a filename, URL, mime type, and stream content. Once the format module has been instantiated, the `JimiReader` or `JimiWriter` maps a user request to the appropriate decoder/encoder to handle the actual image I/O.

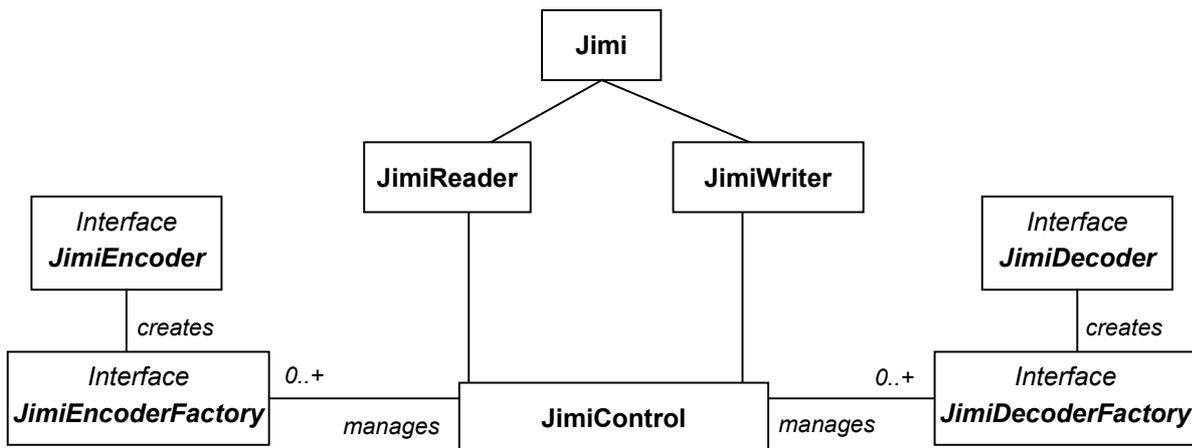


Figure 7.1: Main JIMI classes [based on the API documentation for JIMI, <http://java.sun.com/products/jimi/>]

8

Mobile Interfaces Tag Library (MITL) framework

The browser technology is becoming pervasive not only on desktop computers but also on mobile devices. Current handsets are equipped with different browsers (e.g. Opera, Pocket IE, NetFront, Minimo, etc.) that support various markup languages such as HTML, XHTML, VoiceXML, SALT, or WML. Additionally, devices that are more powerful come with a Java ME environment and guarantee access to device-specific APIs. The introduction of a platform-independent markup language can solve, at least to some extent, the problem of the “tower of Babel” in mobile UI languages. Support of different browsers and markup languages is an issue that can be managed at the user interface level of abstraction – this kind of solution is applied in User Interface Description Languages. An additional challenge in the device-independent authoring is the capture of delivery context. Mobile devices are equipped with screens of different sizes and resolutions, have diverse input mechanisms, support various amounts of colors, fonts and media formats. It may be therefore necessary to display the same content by splitting it into many fragments presented on a different number of screens and to use diverse navigational structures, depending on the platform. Device profiles should be sent as part of the request and should deliver information about the limitations and restrictions of the target device.

Although Java ME is enjoying growing popularity, device-independent approaches that would support the generation of Java ME applications at runtime hardly exist¹¹⁶. Traditional server-based solutions are able to deliver content to thin clients and present it in a browser with the help of some markup language. Alternatively, they can send the data to a Java ME application that was pre-installed on the device when the application requests it. An application may also receive a data push from a predefined IP address. It should be, however, possible to dynamically generate an entire Java ME application on the server from a high-level description language and to send it to a mobile device in response to a request.

The success of mobile Internet depends on the availability and quality of the content that can be accessed from mobile devices. If the users had the possibility to view similar content on mobile devices and on desktop computers, they would be more eager to use their handsets. To avoid the duplication of efforts, developers should have the possibility to reuse existing Internet pages for the generation of mobile pages by extracting and eventually manipulating the presented content [cf. Jank03a].

¹¹⁶ One of the exceptions is the approach for generation of Java ME applications introduced in [Carb+02].

This chapter introduces a library for server-side adaptation called Mobile Interfaces Tag Library (MITL) that automatically adapts the content to the features of mobile appliances and describes a framework that is based on this library [Jank04; Jank05]. The MITL framework¹¹⁷ detects devices characteristics using information included in CC/PP profiles or, alternatively, HTTP headers and delivers data in HTML, WML, XHTML, and Java ME. Section 8.1 provides an overview of design objectives and principles that should be fulfilled by the framework. Section 8.2 describes the evolution of architectures from simple programs to multi-tier architectures. Section 8.3 introduces the architecture of the developed framework and open-source technologies that were used. It focuses on wrappers and existing JSP tag libraries that help to communicate with a database and to parse XML documents as well as on tools for manipulating RSS data feeds. In section 8.4, the developed Mobile Interfaces Tag Library is described. The subsequent section deals with the Integrated Development Environment (IDE) for MITL that facilitates the development of MITL-based documents. Section 8.6 gives some examples of applications developed with the help of the library and section 8.7 presents the results of the evaluation of the developed framework performed by students and developers.

8.1 Design objectives

The main goal of the developed framework is to enable cost-efficient development of highly usable applications for multiple mobile devices. The Mobile Interfaces Tag Library (MITL) framework should provide a markup language and an Integrated Development Environment that would facilitate the device-independent authoring of mobile applications. High software quality should be guaranteed by the fulfillment of the following criteria:

- usability, i.e. understandability, learnability, and operability – the developed markup language and IDE should be easy to understand, learn and apply by end-user programmers and people without extensive programming knowledge.
- time- and resource-related efficiency - the effort required to develop a device-independent application with the help of the MITL framework (in terms of time and needed resources) should be lower than the effort necessary to author the same solution in various markup languages.
- maintainability, i.e. changeability and analyzability – developers with Java EE experience should be able to analyze, change, and extend the existing solution (the markup language and its respective IDE).
- portability – the solution should be platform-independent.

¹¹⁷ The framework was named after the library, since it is its most important component.

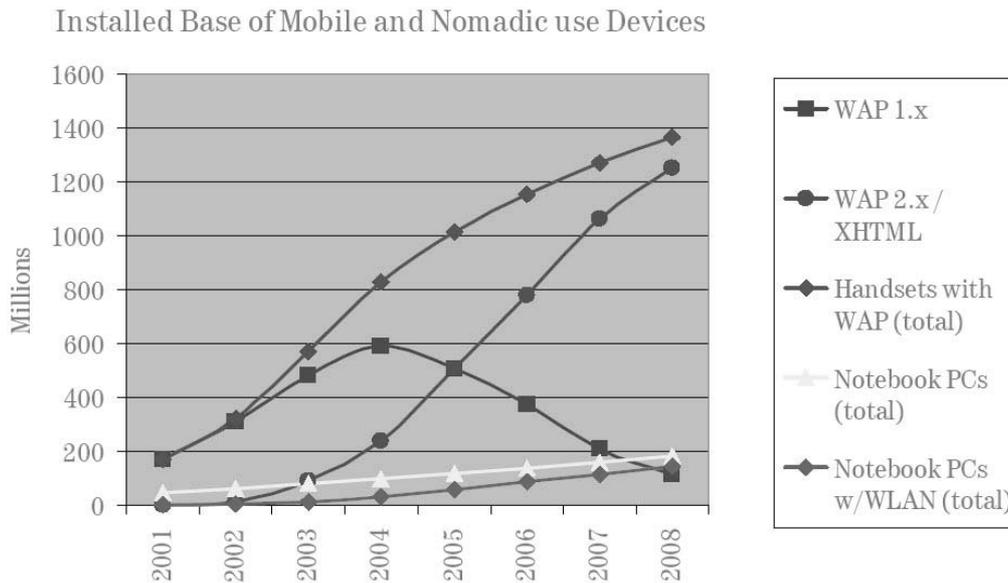


Figure 8.1: Mobile devices supporting WML and XHTML [KoSm04, p. 6]

The adaptation architecture should be influenced not only by general good design practices but also by the design principles and authoring challenges specified by the W3C. The following objectives were identified for the MITL framework:

- affordability, minimization of effort - development of device-independent applications should be cost-efficient and easy to accomplish, support for a variety of devices should be possible without excessive effort.
- support of browser-based and standalone applications, i.e. HTML, XHTML, WML, and Java ME - the WML code should also be produced because the number of handsets that support this standard will remain significant until 2008 (e.g. in 2007 around 200 million devices will still have WML-based browsers, cf. figure 8.1).
- simplicity - simple applications should be developed without great effort, the complexity of development should scale smoothly with the complexity of applications.
- extensibility, open-source - the architecture should be implemented as an open-source, platform-independent, and extensible solution. It should be therefore possible to add support for further languages such as VoiceXML or SALT without the need to change the whole framework, simply by adding new tags to the language or extending existing ones.
- full development cycle - the framework should support full application development cycle without setting any restrictions on data representations or information flow.
- support of device-dependent and device-independent content - it should be possible to include device-dependent and device-independent content in applications.
- support for existing applications (inclusion of existing content/conversion of content) - the authoring technique should provide support for including parts of existing applications during the design or at runtime. It should be possible to generate mobile application by transforming the existing content of

Web pages. RSS feeds and Web requests should also be used as sources of content for device-independent applications.

- exploitation of device capabilities - the framework should be able to access and express the delivery context.
- generation of highly usable target markup - the framework must be able to produce highly usable target markup pages. The usability refers particularly to the ease of interacting with a page. A generated site should be straightforward to understand, well-structured, and easy to navigate.

The last objective includes a set of further requirements with regard to the adaptation of style, layout, content, structure, navigation and application's interactions, particularly:

- navigation support - a developer should be able to implement different navigation methods with minimal effort
- organization - the amount of content displayed on various devices with different presentation techniques should vary
- images conversion - images should be converted to another graphical types or scaled
- integration of application data - author should be able to deliver various presentations of application data depending on the display size and storage capabilities
- aggregation/fragmentation - aggregation of presentation units and decomposition of resources should be possible
- layout and style - different spatial and temporal layouts should be supported for different delivery contexts. The developer should also be able to influence the style of the displayed documents depending on the properties of devices or user preferences.

8.2 Evolution of architectures - from early programs to multi-tier systems

According to a popular definition, software architecture is “a structure or structures of a system that comprise software elements, the externally visible properties of those elements, and the relationships among them” [BaCIKa03, p. 20]. The architecture of a system addresses important quality goals such as security, reliability, usability, modifiability, stability, and real-time performance. It represents design decisions that are most difficult to change [Parn76] and need to be validated against quality goals.

Software development was born around 1949 with the creation of the first stored-program computer, the Cambridge EDSAC. Since then this field experienced a shift almost every decade. The earliest programs were created as binary machine instructions. They were then replaced by a human-readable shorthand for designing programs. The programming shorthand split the tasks between a designer and a coder. The designer structured the program and the coder translated it manually into binary code. In the early 1950s, program subroutines emerged and programmers were able to reuse fragments of programs. This resulted in meaningful productivity improvements. By the late 1950s, automatic programming was announced as the solution to the common programming problems. It allowed

programmers to write software in a high-level language code. This code was then converted into binary machine instructions with the help of another program.

During the 1960s, the number of software development contractors and specialized programs for vertical markets rapidly increased. Software was not given away as part of the hardware platform anymore, it was sold separately. After the introduction of abstract programming interfaces, programs became more portable across hardware platforms. By the end of 1960s, they featured high complexity and had become critical components for many enterprises.

In 1968, a software system was for the first time designed using hierarchical layers [Dijk68]. Around this time, software design was also separated from implementation. Tools, techniques, and modeling languages began to emerge. The foundations of software architectures were laid out by Parnas. In 1972, he published a famous paper on modularity in the design of systems and introduced the concept of information hiding [Parn72]. According to this idea, each module should encapsulate (“hide”) an essential design decision and therefore future design changes would affect only one module and not the entire software.

In the 1980s, object-oriented programming and graphical user interfaces (GUIs) revolutionized the world of applications. In early 1990s, the term software architecture began to show up in the literature. Terminal servers were replaced by networks of powerful personal computers; client/server architectures and the concept of layers emerged. Most applications have three major layers: presentation layer, business logic layer, and services layer. The presentation layer provides the interaction between a machine and a user (user interface). It handles input from the keyboard, mouse, or other devices and displays the output to the user. The application or business logic contains the business rules typical for a particular application. The last layer provides various services such as file services, print services, communications services, and database services [cf. Brit00].

The number of tiers in an architecture is determined by the degree of integration between the three aforementioned layers. In one-tier applications, all layers are tightly connected and the presentation layer knows about the database structure. In a two-tier design (the so-called client/server architecture), database services are separated from the application and are placed on a separate machine. The presentation and business logic layers remain intertwined, and they both possess knowledge of the database organization. The presentation layer is positioned on a client and the business logic tends to be split arbitrarily between the client and the server tier. This split leads to two variants of the architecture referred to as a fat client and a thin client architecture. In the thin client variant, the business logic remains on the server whereas in the two-tier architecture with a fat client most of the business logic resides on a powerful client.

In three-tier design, business logic becomes a distinct service, and is placed on a separate machine that is called the application server. The presentation layer usually does not have knowledge of the database. It communicates with the application server using a predefined message strategy. Three-tier architectures can further be extended to multi-tier (or n-tier) architectures. In such architectures, additional tiers and layers are introduced. For example, business logic layer may communicate with services layer with the help of a further layer called data access layer. This layer is responsible for

processing business logic layer's requests and querying/modifying a database. Similarly, the presentation layer may be split into the presentation logic and presentation elements (GUI). Separate presentation logic is particularly important if similar data is delivered to different devices supporting various markups. Multi-tier architectures emerge if the mentioned layers (data access layer, presentation logic layer, presentation layer, etc.) are directly associated with separate tiers. The major reason for the introduction of multi layered design was the possibility to enclose application parts into layers that can be easily modified or exchanged without affecting the rest of the architecture.

8.3 Framework's architecture and open-source technologies used

8.3.1 General architecture

The developed framework consists of four layers depicted in figure 8.2: data layer, business logic layer, presentation logic layer, and presentation layer. The data layer encompasses information stored in different databases (e.g. Oracle, MS SQL Server, PostgreSQL, etc.) and the content from existing Web pages (in HTML, XHTML, or XML). The business logic layer consists of an advanced wrapper and JSP Standard Tag Library (JSTL). This layer manages the data access for the presentation layer and keeps track of session data. The wrapper is responsible for the extraction and manipulation of content included in Web pages. JSTL is in charge of XML processing, database interactions, and conditional logic. Instead of a direct communication with a database with the help of the JSTL tag library, data access may also be realized by introducing Java Beans to the business logic layer. The choice is, however, left to the developers. The presentation logic layer contains JavaServer Pages with MITL tags. When a JSP engine recognizes a MITL tag, it invokes an appropriate Java class associated with this tag and generates some output. Depending on device properties, the produced code is in the XHTML, HTML, WML, or Java ME format. The presentation layer consists of mobile devices (including notebooks) equipped with wireless browsers or supporting Java ME.

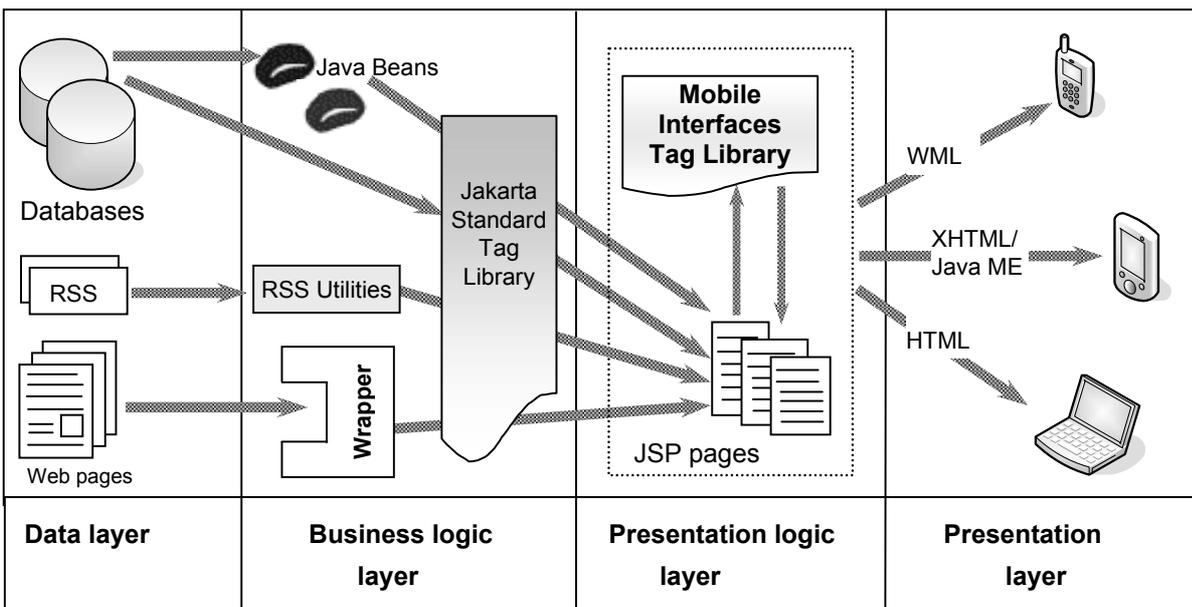


Figure 8.2: Architecture of MITL framework

The advantages of this architecture are as follows:

- changes to the application logic or to the user interface can be undertaken without influencing the rest of the architecture. Therefore the applications can evolve straightforwardly and meet new requirements
- network bottlenecks can be minimized because the application layer does not have to transmit the bulk of additional data to the client
- the client is insulated from database and network operations, it can access information without any knowledge about the location of data
- the developers and Web designers can participate in the development process and focus on their specializations
- it is possible to create device-independent solutions without any programming knowledge.

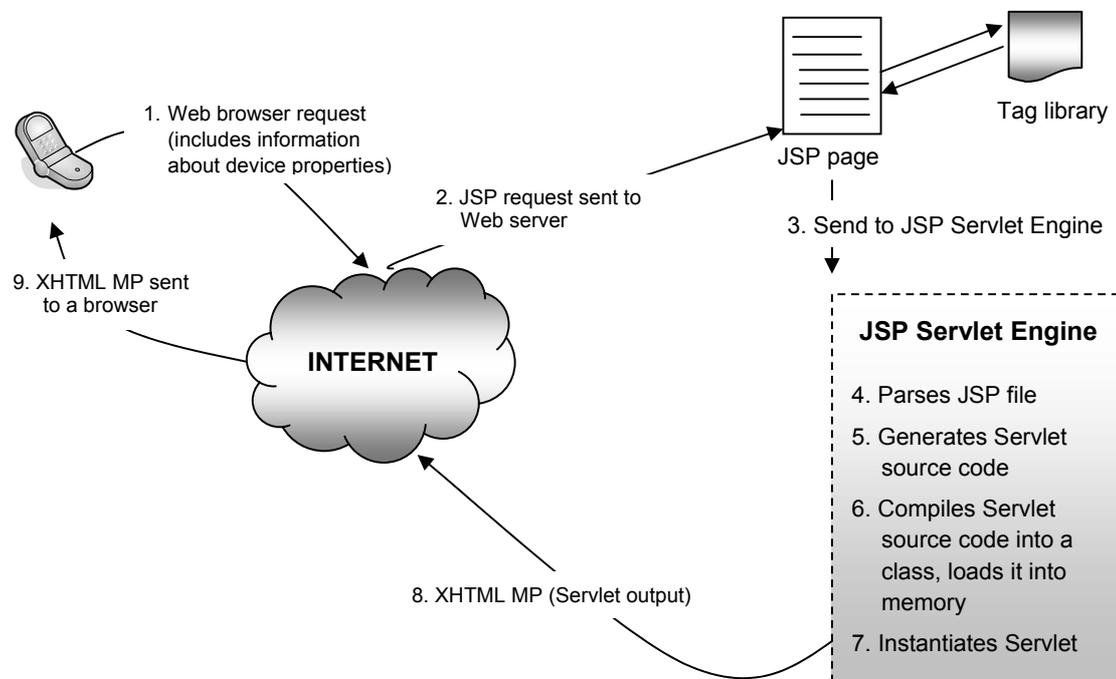


Figure 8.3: Request processing in MITL framework

The processing of requests in the framework is performed in the following way (cf. figure 8.3):

- 1) The browser sends a request for a unique URL (JSP page) to a Web Server running a JSP container (i.e. Tomcat). As part of the request, information about a device profile in the form of CC/PP or HTTP header is also delivered.
- 2) The request is intercepted and the Web server invokes a particular JSP page. This JSP page consists of tags specified in the Mobile Interfaces Tag Library that are responsible for presentation, and tags from the JSP Standard Tag Library that are in charge of business logic. It can also contain some markup tags.

- 3-7) The JSP page is sent to the JSP Servlet Engine. The engine parses the JSP file and generates a Servlet source code from it. Each time the engine encounters a known tag, it invokes a Java class associated with it. Subsequently, the Servlet source code is compiled into a Java class, the class is loaded into memory and instantiated. If a JSP page has already possessed a servlet representation, the page is not compiled, but the existing Servlet is invoked.
- 8) Depending on the device features extracted from the CC/PP profile or HTTP headers, the Servlet generates some output in the WML, XHTML or HTML format or produces a Java ME application.
- 9) The output is sent to the browser and displayed in it. If a Java ME application was generated by the JSP page, the user obtains a JAD file first. In the next step, he/she is able to download the respective JAR file and can run the application.

8.3.2 JSP tag libraries

JavaServer Pages combine some kind of markup language (usually HTML) with Java code. They help to generate easy-to-read graphical interface code and encourage the separation between the presentation layer and the application logic layer. Custom tags, also known as tag extensions or custom actions, can encapsulate the presentation/application logic into independent Java classes, centralizing the implementation of presentation or application logic, facilitating maintainability and the division of tasks between Web page designers and Java developers. The MITL framework is based on two types of tag libraries (collections of tags): a standard third-party library (JSTL) and the MITL tag library developed by the author of this thesis.

Custom tags have access to the implicit JSP objects such as `pageContext`, `request`, `response`, `page`, `application`, `session`, `config` and `JSPWriter` objects. Tags can write directly into the output stream of JSP and are able to use the JSP exception handling mechanism. JSP actions¹¹⁸ can be nested and it is possible to pass object references between actions. A tag library is a JAR file that contains a tag library descriptor (`taglib.tld`) file, appropriate Java classes and other resources required to implement tags such as images or text files [cf. Rock01, p. 310-312].

Tag libraries have various advantages. They enable Web page designers to build complicated JSP pages without any Java programming knowledge. The way of creating pages is consistent and easy-to-follow. A tag can include a body and a list of arguments and resembles commonly known HTML tags. All tasks that are usually included in Java scriptlets such as conditional statements or accessing a database can be abstracted in the tags.

For device-independent content creation, tags are particularly useful. They can encapsulate the generation of content tagged with different markups depending on the device features extracted from

¹¹⁸ Actions in JSP range from printing a script expression or executing a scriptlet, to creating and storing a Java Bean.

CC/PP profiles or HTTP headers. Content authors can create device-independent pages without worrying about screen size of a device, supported markup languages, or font types. They do not have to check the correctness of the syntax of each generated markup. If the generated output is not satisfactory or new languages are to be supported, Java developers can simply add new tags or change the business/presentation logic implemented in a library. The subsequent paragraphs describe the implementation of a sample tag library.

Developing a custom tag library

Tag libraries have to be imported into a JSP page before they can be used. This can be accomplished with the directive `<%@ taglib %>` that requires two attributes: `uri` and `prefix`. A Universal Resource Identifier (URI) is associated with a tag library and differentiates it from other tag libraries. A tag library also possesses an XML-like namespace prefix that distinguishes all the tags belonging to a particular tag library. For instance, the directive: `<%@ taglib uri=http://www.xyz.com/myLib prefix="mylib"%>` imports the tag library identified by the URI `http://www.xyz.com/myLib` using the prefix `mylib`. After this directive appears in a page, all tags from the library can be used with the `mylib` prefix, e.g. `<mylib:sometag>`.

Tag libraries are developed in Java using the so-called JSP's tag extension API. With the help of this Application Programming Interface, programmers can write Java classes that implement custom JSP tags, the so-called tag handlers. For example, if a developer writes a Java class `RandomVal` that is a part of the `mytagext` package (cf. listing 8.1), he/she can associate it with a particular tag, e.g. `<mytagext:randomVal>` and can use it in a JSP page. All Java classes included in one library can be assembled in a JAR file and described in an XML-formatted tag library descriptor (TLD) configuration file (cf. listing 8.2). Subsequently, a reference to the tag library should be added to the Web application deployment descriptor (cf. listing 8.3).

```
1. package mytagext;
2. import java.io.*;
3. import javax.servlet.jsp.*;
4. import javax.servlet.jsp.tagext.*;
5.
6. public class RandomVal extends BodyTagSupport {
7.     public int doStartTag() throws JspException {
8.         return SKIP_BODY;
9.     }
10.    public int doEndTag() throws JspException {
11.        int randVal = (int) (Math.random()* 100);
12.        try {
13.            pageContext.getOut().write(randValue);
14.        } catch (IOException e){
15.            throw new JspException(e.getMessage());
16.        }
17.        return EVAL_PAGE;
18.    }
19. }
```

Listing 8.1: *RandomVal* tag handler

```

1. <?xml version="1.0" encoding="ISO-8859-1" ?>
2. <!DOCTYPE taglib PUBLIC
3. "-//Sun Microsystems, Inc./DTD JSP Tag Library 1.2//EN"
4. "http://java.sun.com/dtd/Web-jsptaglibrary_1_2.dtd">
5. <taglib>
6.     <tlib-version>1.0</tlib-version>
7.     <jsp-version>1.2</jsp-version>
8.     <short-name>useful funcs</short-name>
9.     <description>A library with some useful functionality</description>
10.
11.     <tag>
12.         <name>randomVal</name>
13.         <tag-class>mytagext.RandomVal</tag-class>
14.         <body-content>empty</body-content>
15.         <description>Generates a random number</description>
16.     </tag>
17.     [...]
18. </taglib>

```

Listing 8.2: TLD file for the *RandomVal* class

```

1. </web-app>
2. <taglib>
3.     <taglib-uri>http://www.someaddress.com/tagext</taglib-uri>
4.     <taglib-location>/WEB-INF/mytagext.tld</taglib-location>
5. </taglib>
6. </web-app>

```

Listing 8.3: Web application deployment descriptor

Tags can possess a body and attributes and have the following general structure:

```

<tagname attributename="attrvalue" otherattributename="otherattrvalue">
    Tag's body.
</tagname>

```

The information that is necessary for the JSP runtime to verify the structure of a tag and to process it is contained in the tag library descriptor (TLD). The elements that can be used in such descriptors are presented in table 8.1. The tag library descriptor from listing 8.2 specifies the name of the tag that will be used in JSP pages (`randomVal`, line 12 of listing 8.2), provides a reference to the appropriate Java class (`mytagext.RandomVal`, line 13) and informs the JSP runtime environment that the tag does not have a body (line 14 of listing 8.2).

All tags must obey a special API. They have to implement the `Tag` interface that defines all the methods the JSP runtime engine calls in order to execute a tag and have a strict lifecycle. The most important methods of the `Tag` interface are summarized in table 8.2. The `BodyTag` interface has to be implemented by all tags that possess a body that should be processed. Since there is usually no need to implement all the methods from the `BodyTag` or `Tag` interfaces, the tag API offers a standard basic implementation for both interfaces: `TagSupport` and `BodyTagSupport`. These classes implement all the mandatory tag properties (e.g. `parent`, `pageContext`) and support the lifecycle methods by

providing some default values for the basic tag methods (`doStartTag()`, `doEndTag()`, or `doAfterBody()`).

Element name	Description	
attribute	A tag can have many attributes or none.	
subelements of attributes	name	The name of the attribute.
	required	A <code>true/false</code> value that specifies whether the attribute is required.
	rtexprvalue	Allows RT expressions to be used as a value of an attribute. RT expressions are expressions such as <code><%=request.getMethod()%></code> .
bodycontent	Specifies whether a tag has a body or is empty. If JSP code can be contained in the body, the value "jsp" should be used.	
description	A textual description of a tag's functionality.	
name	The name of a tag to which the JSP code will refer to.	
tag-class	The name of a class that implements the tag.	

Table 8.1: Elements in tag library descriptor [Baye03, pp. 363-365]

Method name	Description
<code>setPageContext(PageContext pc)</code> Tag Interface	Called by the JSP runtime to set the <code>PageContext</code> for a tag. This gives the tag handler a reference to all the objects associated with the page.
<code>setParent(Tag t)</code> Tag Interface	Called by the JSP runtime to pass a tag handler a reference to its parent tag (if it has one).
<code>getParent()</code> Tag Interface	Returns a <code>Tag</code> instance that is the parent of this tag.
<code>doStartTag()</code> Tag Interface	Called by the JSP runtime to prompt the tag handler to do its work and to indicate (via returned value) what the engine runtime should do next.
<code>doEndTag()</code> Tag Interface	Called by the JSP runtime when it reaches the end mark of a tag to allow it to do additional work and to indicate (via returned value) what to do next.
<code>release()</code> Tag Interface	Called by the JSP runtime to prompt the tag handler to perform the cleanup before the tag is reused.
<code>setBodyContent(BodyContent bc)</code> BodyTag Interface	Called by the JSP runtime to set a <code>BodyContent</code> object for this tag. This gives the tag handler access to its processed body.
<code>doInitBody()</code> Tag Interface	Called by the JSP runtime to prompt the tag handler to perform any needed initialization before processing its body.
<code>doAfterBody()</code> BodyTag Interface	Called by the JSP runtime after it reads in and processes a tag's body to prompt the tag handler to perform any inspection or modification of the processed body.

Table 8.2: Tag and BodyTag interface methods [ShChRy01, p.85]

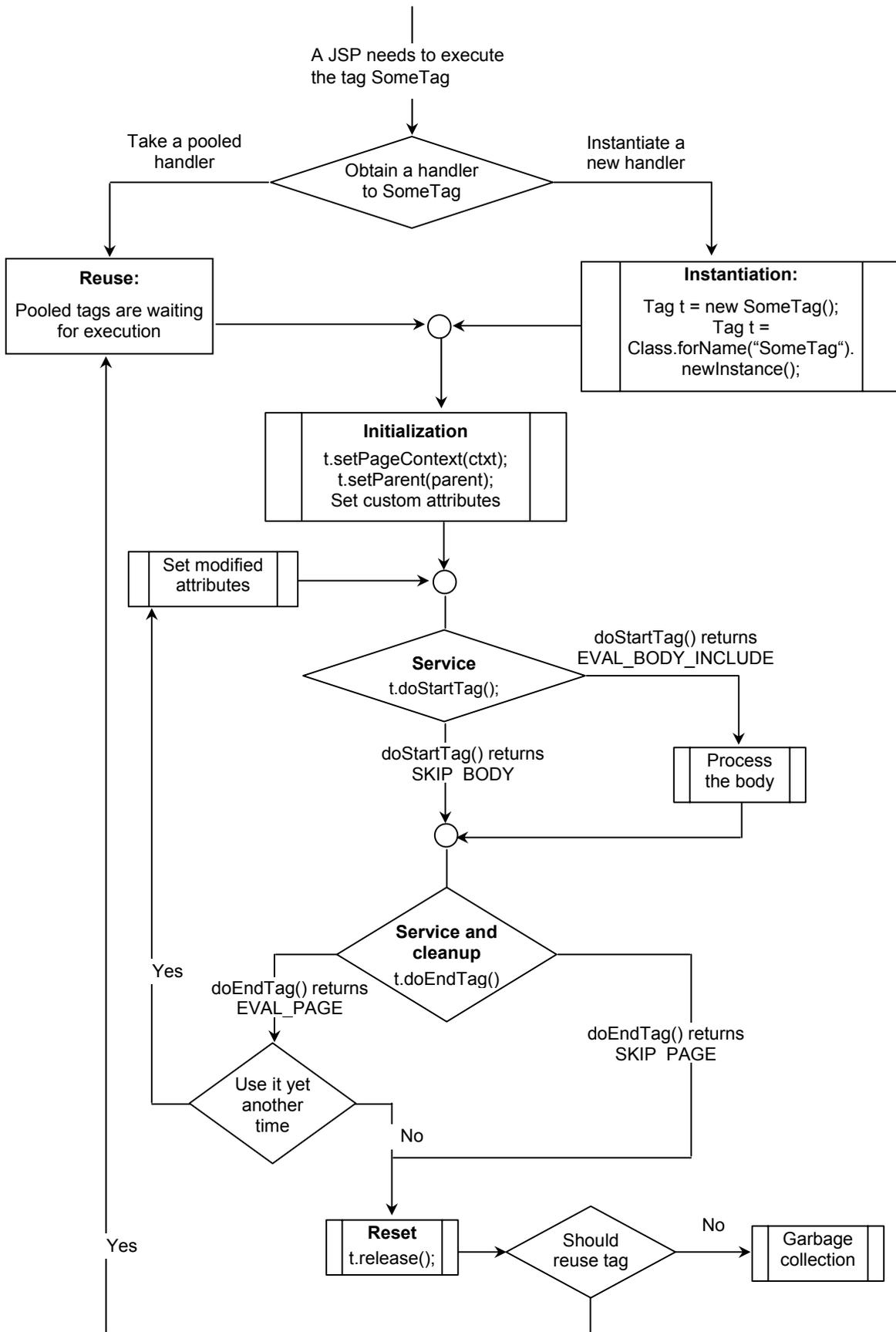


Figure 8.4: Tag lifecycle [ShChRy01, p. 91]

Each tag has a certain lifecycle that can be divided into the following phases (cf. figure 8.4):

- instantiation - takes place when the JSP runtime needs a new copy of a tag handler
- initialization - before a tag can be used by the JSP runtime, it has to be initialized
- service - certain tag functionality is executed
- cleanup - the tag cleans itself from a state generated during the service
- reuse - the tag handler can be reused in further tag executions
- garbage collection - the tag is garbage collected and can no longer be used.

When the JSP runtime needs to execute a tag, an instance of the tag handler is necessary. A JSP page can obtain the tag handler from a pool of already instantiated tag handlers or it can instantiate a new instance. Tags may be instantiated in two ways: by calling the default constructor (e.g. `Tag t = new SomeTag();`) or by using the `Class.forName()` method (`Tag t = Class.forName("SomeTag").newInstance();`).

After the instantiation of the tag, the JSP runtime initializes the handler by setting its properties such as `pageContext`, `parent`, or custom tag attributes. Subsequently, it starts the execution of the tag by calling the `doStartTag()` method. This method encapsulates the business logic of a tag and can return the `SKIP_BODY` or `EVAL_BODY_INCLUDE` values. The `SKIP_BODY` value informs the JSP runtime to skip the tag's body, while the `EVAL_BODY_INCLUDE` value instructs the runtime to process the body. Finally, the `doEndTag()` method is called. It can contain some additional functionality and returns the `SKIP_PAGE` or `EVAL_PAGE` value. `SKIP_PAGE` means that the processing of the remaining parts of a page will be aborted at this place and `EVAL_PAGE` signifies that the runtime can proceed with the processing.

After the `doEndTag()` method is invoked, the tag should be left in a state that would enable its reuse. Therefore, in the `doEndTag()` method all the instance variables (e.g. database connections, lists, HashTables, etc.) should be cleaned and reinitialized to the values from the beginning. This step is necessary because tags are often pooled and reused. Therefore, after the `release()` method the JSP runtime expects the state of a tag handler to be identical with the state after the execution of an empty constructor and uses the existing instance instead of creating a new one.

The described lifecycle differs for the `BodyTag` and `IterationTag`. The `BodyTag` interface offers a possibility to get access to the body of a tag and to process its content. The `BodyTag` interface introduces three additional methods for processing the body and a new return value `EVAL_BODY_TAG`. The `EVAL_BODY_TAG` return code means that the JSP runtime should inspect and eventually modify the tag's body. The new methods added in the `BodyTag` interface are: `setBodyContent()`, `doInitBody()`, and `doAfterBody()`. The `setBodyContent()` method passes a reference to the tag's body and accepts as the argument an instance of the `BodyContent` class. This class has methods for reading the body and writing some changes to it. The `doInitBody()` method allows any initializations (e.g. setting variables, creating new objects) that would be required before the evaluation of the body. The `doAfterBody()` method makes it possible to change the body of a tag. This method can be invoked many times, for example, until a certain condition is satisfied.

The lifecycle of the `BodyTag` is similar to the lifecycle of a `Tag`. The changes are highlighted in gray in figure 8.6 and occur mainly in the body handling phase that is part of the service phase. When the `doStartTag()` method returns the `EVAL_BODY_TAG` return code, the `setBodyContent()` and `doInitBody()` methods are called. The obtained body can be manipulated in the `doAfterBody()` method. Should the body be processed many times, the method returns the `EVAL_BODY_TAG`, otherwise it returns the `SKIP_BODY` value and the `doEndTag()` method is invoked.

Since the body often needs to be evaluated several times, in the JSP 1.2 the `IterationTag` interface was introduced. This interface allows to repeat the evaluation of the tag's body, but does not require to set a `BodyContent` object. The interface also adds two new return codes: `EVAL_BODY_BUFFERED` and `EVAL_BODY_AGAIN`. In JSP 1.2 the `EVAL_BODY_TAG` is depreciated in the `doStartTag()` method and is replaced with the `EVAL_BODY_BUFFERED` constant. The `EVAL_BODY_BUFFERED` preserves the value of the old `EVAL_BODY_TAG` (it is set to 2), allowing tags built according to the older version of JSP to run in newer environment. With the introduction of `IterationTag` in JSP 1.2, tags can ask the JSP environment to re-evaluate their body, even if these tags are not `BodyTags`¹¹⁹. In the `doAfterBody()` method the `EVAL_BODY_TAG` can be replaced by the `EVAL_BODY_AGAIN` constant. If this value is returned, the JSP runtime re-evaluates the body once again. The lifecycle of the `IterationTag` is presented in figure 8.5. The return codes are summarized in table 8.3.

Return values for interface methods	Description
<code>EVAL_BODY_INCLUDE (doStartTag())</code>	Content contained within the tag is included in the output of a page.
<code>SKIP_BODY (doStartTag())</code>	Content contained within the tag is ignored.
<code>EVAL_BODY_BUFFERED (doStartTag())</code>	Content of the body is re-evaluated.
<code>EVAL_PAGE (doEndTag())</code>	Content after the tag is processed normally.
<code>SKIP_PAGE (doEndTag())</code>	Content after the tag is skipped.
<code>EVAL_BODY_TAG (doAfterBody(), (doStartTag()))</code>	Content contained within the tag is evaluated by the <code>doAfterBody</code> method again, presumably until a specific condition is met.
<code>SKIP_BODY (doAfterBody())</code>	Processing of the tag is complete.
<code>EVAL_BODY_AGAIN (doAfterBody())</code>	Content of the body is re-evaluated.

Table 8.3: Return codes for tags [cf. *ShChRy01*, pp. 81-106]

¹¹⁹ This is possible because the `BodyTag` is an extension of the `IterationTag`.

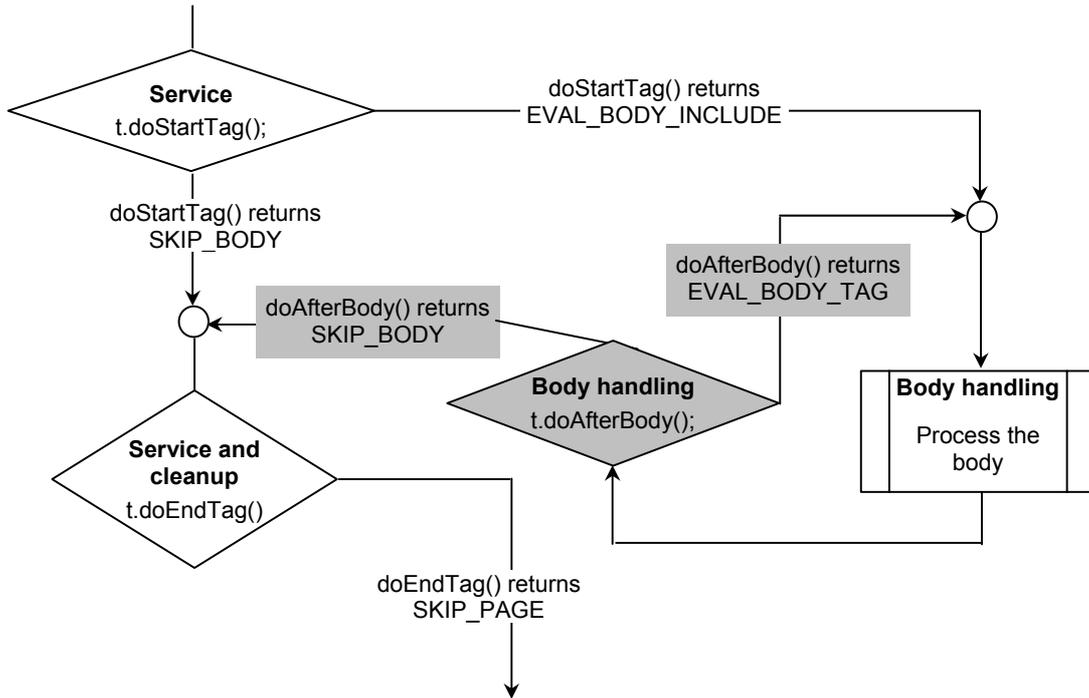


Figure 8.5: IterationTag lifecycle [ShChRy01, p. 104]

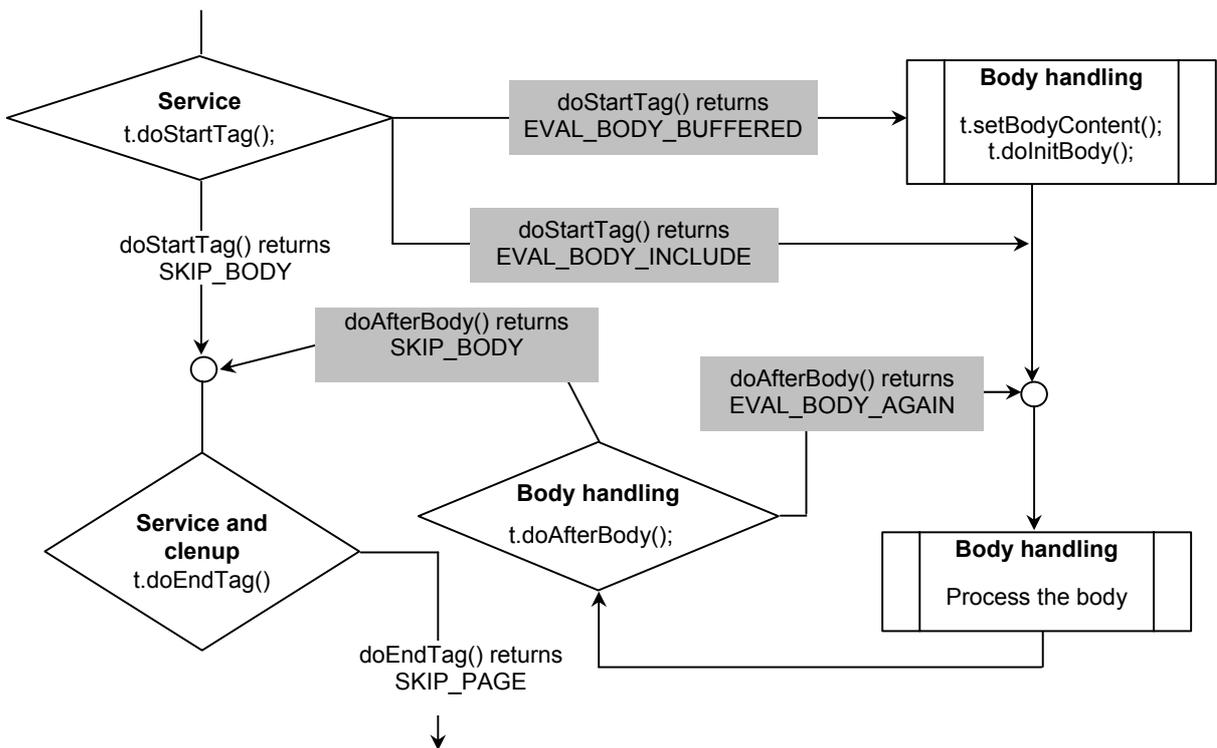


Figure 8.6: BodyTag lifecycle (JSP 1.2) [ShChRy01, p. 98]

8.3.3 JavaServer Pages Standard Tag Library

JavaServer Pages Standard Tag Library (JSTL) [ShChRy01; Baye03] is a collection of tags developed by the Apache Software as part of the Jakarta Taglibs project¹²⁰. The JSTL tag library consists of the following four libraries: core library, processing library, internationalization and formatting library, and database access library. The core library allows modifying data in memory and offers access to it. It also includes some tags for standard actions like iterations or support for conditional logic. The XML library makes parsing of XML documents possible. Certain parts of the XML documents may be extracted or manipulated and XML files may be printed out. The formatting and internationalization library supports the internationalization of applications and formatting of numbers and dates. Last but not least, the database access library helps to read data from a database and to write it to the database. The libraries together with their suggested prefixes and sample tags are summarized in table 8.4.

JSTL tag library	Suggested prefix	URI	Exemplary tag
Core library (iteration, conditions)	c	http://java.sun.com/jstl/core	<c:forEach>
XML processing library	x	http://java.sun.com/jstl/xml	<x:forEach>
Internationalization (i18n) and formatting	fmt	http://java.sun.com/jstl/fmt	<fmt: formatDate>
Database (SQL) access	sql	http://java.sun.com/jstl/sql	<sql:query>

Table 8.4: JSP Standard Tag Library [Heat03, p. 18]

Core JSTL library

The core JSTL library supports generation of output, management of variables, flow control and iterations, text imports, and URL manipulation. Table 8.5 presents the basic tags from this library and describes shortly their meaning. These tags can furthermore possess different attributes.

Tag	Description
<c:out>	Writing data.
<c:set>	Saving data into memory.
<c:remove>	Deleting data.
<c:catch>	Error handling.
<c:if>	Support for a simple yes-or-no condition.
<c:forEach>	Iteration.
<c:forEachTokens>	Looping through Strings (Strings are broken into fragments called tokens).
<c:import>	Inclusion of text.

¹²⁰ <http://jakarta.apache.org/taglibs/>.

Tag	Description
<c:url>	Printing and formatting a URL.
<c:redirect>.	URL redirection.
<c:param>	Passing Strings from the source page to the target page.
<c:choose>, <c:when>, <c:otherwise>	Support for multiple exclusive conditions.

Table 8.5: Basic tags from core JSTL library [cf. Baye03, pp. 43-138]

In the example presented in listing 8.4, the JSP page prints out one of the titles: “Dr.”, “Ms.”, “Mr.” or “-” (title not specified). The tags check the education of a user and his/her gender and produce appropriate output. The education is checked at the beginning, since it is not dependent on the gender.

```

1. <c:choose>
2.   <c:when test="\${user.education=='doctorate'}"> Dr. </c:when>
3.   <c:when test="\${user.gender=='female'}"> Ms. </c:when>
4.   <c:when test="\${user.gender=='male'}"> Mr. </c:when>
5.   <c:otherwise> - </c:otherwise>
6. </c:choose>
7. <c:out value="\${user.name}"/>

```

Listing 8.4: Exemplary usage of core JSTL library

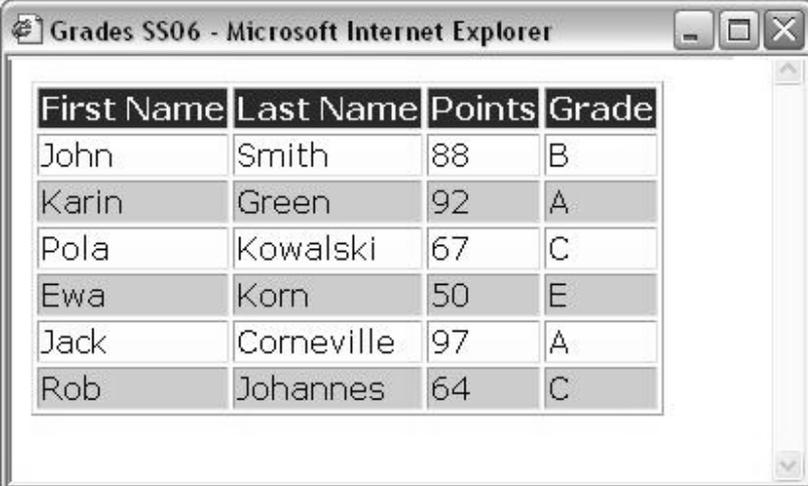
XML processing

The XML processing tag library allows a developer to perform various operations on XML data. It supports parsing of XML documents, selection of XML fragments, flow control based on XML, and XSLT transformations. XPath is used for selecting XML fragments and comparison of XML data. During the parsing process, the syntax of text is analyzed and data is turned into a useful representation, i.e. a format that can be handled with XSLT, XPath or other XML-related technology. Table 8.6 presents the most useful tags from the XML processing tag library.

Tag	Description
<x:parse>	Parses an XML document.
<x:out>	This tag evaluates and prints out the string value of an XPath expression.
<x:set>	Saves a scoped variable representing XML data, or representing data pulled from an XML document.
<x:if>, <x:choose>, <x:when>, <x:otherwise>	Tags that support conditional logic in XML processing.
<x:forEach>	This tag is used for looping through XML nodes.
<x:transform>	Enables the conduction of XSLT transformations from within a JSP page.
<x:param>	This tag can set a parameter in an XSLT style sheet.

Table 8.6: Tags from XML processing library [cf. Heat03, 195-216]

In the example presented in listings 8.5-8.7, the XML file “students.xml” is transformed into HTML with the help of the “transform.xsl” style sheet. The transformation is performed by the tags within the JSP page. At the beginning, both files are imported into the JSP page (lines 3-4 in listing 8.7). Subsequently, the XML data is converted into HTML by applying the `<x:transform>` tag. XPath expressions are used to extract the relevant fragments of data and to present them in the form of a table as displayed in figure 8.7.



First Name	Last Name	Points	Grade
John	Smith	88	B
Karin	Green	92	A
Pola	Kowalski	67	C
Ewa	Korn	50	E
Jack	Corneville	97	A
Rob	Johannes	64	C

Figure 8.7: Result of transformation with XML processing library

```

1. <?xml version="1.0" encoding="ISO-8859-1"?>
2. <students>
3.   <student id="1">
4.     <name>
5.       <first>John</first>
6.       <last>Smith</last>
7.     </name>
8.     <grade>
9.       <points>88</points>
10.      <letter>B</letter>
11.    </grade>
12.  </student>
13.  [...]
14.  <student id="6">
15.    <name>
16.      <first>Rob</first>
17.      <last>Johannes</last>
18.    </name>
19.    <grade>
20.      <points>64</points>
21.      <letter>C</letter>
22.    </grade>
23.  </student>
24. </students>

```

Listing 8.5: XML file (students.xml)

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <xsl:stylesheet version="2.0"
3. xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4. xmlns:xs="http://www.w3.org/2001/XMLSchema"
5. xmlns:fn="http://www.w3.org/2004/07/xpath-functions"
6. xmlns:xdt="http://www.w3.org/2004/07/xpath-datatypes">
7. <xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>
8. <xsl:template match="students">
9. <html>
10. <head>
11. <title>Grades SS06</title>
12. </head>
13. <body>
14. <font face="Verdana">
15. <table border="1">
16. <tr bgcolor="003366">
17. <th><font color="white">First Name</font></th>
18. <th><font color="white">Last Name</font></th>
19. <th><font color="white">Points</font></th>
20. <th><font color="white">Grade</font></th>
21. </tr>
22. <xsl:for-each select="student">
23. <tr>
24. <xsl:if test="position() mod 2=0">
25. <xsl:attribute name="bgcolor">lightblue</xsl:attribute>
26. </xsl:if>
27. <td><xsl:value-of select="name/first"/></td>
28. <td><xsl:value-of select="name/last"/></td>
29. <td><xsl:value-of select="grade/points"/></td>
30. <td><xsl:value-of select="grade/letter"/></td>
31. </tr>
32. </xsl:for-each>
33. </table>
34. </font>
35. </body>
36. </html>
37. </xsl:template>
38. </xsl:stylesheet>

```

Listing 8.6: XSLT file (transform.xsl)

```

1. <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
2. <%@ taglib uri="http://java.sun.com/jstl/xml" prefix="x" %>
3. <c:import var="xml" url="students.xml" />
4. <c:import var="xslt" url="transform.xsl" />
5. <x:transform xml="{xml}" xslt="{xslt}" />

```

Listing 8.7: Example of a JSP page with tags from XML processing library

Internationalization and formatting

JSP Standard Tag Library supports internationalization (i18n) of applications and localized formatting. Language-specific strings can be stored in the so-called Java resource bundles, from which the application can read the appropriate language version and display it. JSTL offers tags for the

specification of new default locales, the preparation of resource bundles and displaying localized messages. JSTL also provides tags to read and display formatted dates and numbers. Additionally, a developer has the possibility to adjust the time zones used for reading and writing dates. Table 8.7 provides an overview of the existing i18n and formatting tags.

Tag	Description
<code><fmt:bundle></code>	This tag is applied to load the specified bundle and use it as the default bundle.
<code><fmt:formatDate></code>	The tag is used to format dates in a variety of standard forms.
<code><fmt:formatNumber></code>	The tag is used to format numbers in a variety of standard forms.
<code><fmt:message></code>	The tag is used to insert multilingual text into JSP pages.
<code><fmt:param></code>	The tag is used to specify parameters for messages.
<code><fmt:parseDate></code>	The tag is used to transform a string into a Java date.
<code><fmt:parseNumber></code>	The tag is used to parse strings into numeric values.
<code><fmt:requestEncoding></code>	The tag is used to specify the encoding method used in the form that posted values to the current page.
<code><fmt:setBundle></code>	The tag is used to load a Java resource bundle into a JSTL scoped variable.
<code><fmt:setLocale></code>	The tag is used to specify the current locale for a Web application.
<code><fmt:setTimeZone></code>	The tag is used to specify the time zone for a Web application.
<code><fmt:timeZone></code>	The tag is used to specify a time zone that will be used by all tags inside the body of the <code><fmt:timeZone></code> tag.

Table 8.7: Basic tags from formatting library [cf. Baye03, pp. 215-248]

8.3.4 Database access

Information displayed on heterogeneous devices may originate from different data sources such as the World Wide Web or Database Management Systems (DBMS). Content can be extracted from different types of databases, can be stored in plain text files, in the form of HTML, XML or in formats established for Semantic Web like Really Simple Syndication (RSS 2.0) [RAB02]. A framework for automated content adaptation should provide common mechanisms for accessing the information and eventually transforming them to a format that would be easy to explore, manipulate, and convert. Information from diverse databases (e.g. Oracle, Access, MySQL, MS SQL Server, Postgress, etc.) can be retrieved using the Java Database Connectivity (JDBC) mechanism. Instead of using JDBC API directly, a DBTags library that encapsulates the functionality of this API can be applied.

Java Database Connectivity

The Java Database Connectivity concept relies on the database drivers that are adapted to specific database management system. The components responsible for data retrieval may differ across Java EE frameworks but they are based on the JDBC API. It is the industry standard for developing database-independent access and communication with wide range of databases in Java. The API consists of classes for establishing connection with a database, accessing any tabular data source,

sending SQL statements, and processing the retrieved results. The JDBC API is included in the Java Platform, Standard Edition (Java SE) and the Java Platform, Enterprise Edition (Java EE).

The JDBC API includes two sets of interfaces - for application and for driver writers. Applications or applets can access databases using four methods. They can apply pure Java JDBC technology-based drivers, use database drivers for database middleware, and exploit ODBC drivers together with existing database client libraries with the help of JDBC-ODBC bridge or a native API driver, partially Java-enabled. All methods are illustrated in figure 8.8.

Pure Java drivers convert JDBC calls into the network protocol used by Database Management Systems (DBMSs) and allow for direct calls from the client machine to the database server. Pure Java drivers for database middleware translate JDBC calls into the middleware vendor's protocol. Subsequently, the calls are translated to a database protocol by a middleware server. The middleware is a connector to heterogeneous databases. The combination of the JDBC-ODBC bridge and ODBC drivers provides access to databases via ODBC drivers and can be used if no JDBC drivers are available. A native API driver converts JDBC calls into calls on the client API for various DBMS. Independently of the access method used, JDBC mechanism enables communication with any type of database and can be extended to new DBMSs.

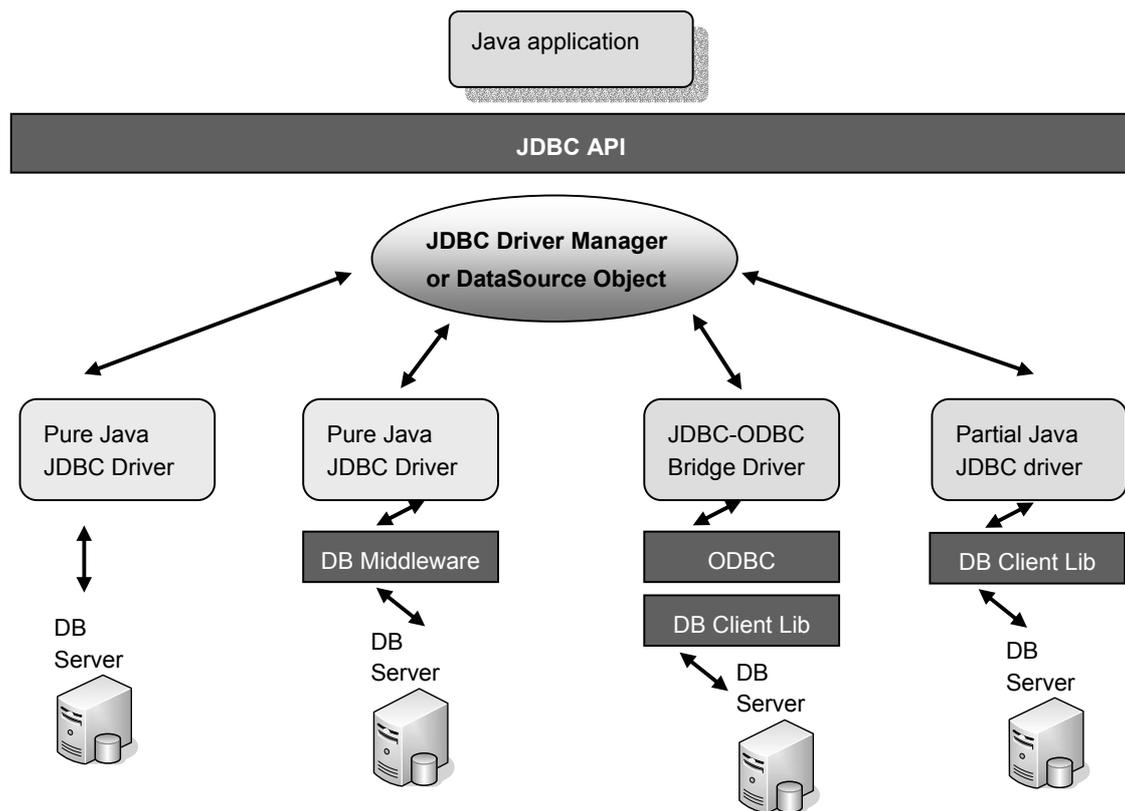


Figure 8.8: JDBC mechanisms for communication with databases [<http://java.sun.com/products/jdbc/overview.html>]

DBTags library

The DBTags library supports typical database operations such as database queries, updates, and transactions (set of database operations). It is very useful for small and medium-size applications, but not suitable for large applications with a large number of users because it does not support connection pooling¹²¹. Furthermore, in typical multi-tier applications database access is usually kept out of JSP pages in components such as JavaBeans or Enterprise Java Beans. However, in small projects the DBTags library may be applied for retrieving data or updating the database. Table 8.8 provides an overview of the supported tags.

Tag	Description
<code><sql:param></code> , <code><sql:dateParam></code>	These tags allow the specification of parameters for SQL queries. If a parameter is a date, the <code><sql:dateParam></code> tag should be used, otherwise the <code><sql:param></code> tag can be applied.
<code><sql:query></code>	This tag is used to submit an SQL query to a database. It returns a set of records through which a user can iterate with the <code><c:forEach></code> tag.
<code><sql:setDataSource></code>	This tag is used to set up a database and to open a connection to a data source. The developer has to know the name of a driver and the URL of the data source.
<code><sql:transaction></code>	This tag is used to group database operations so that they succeed or fail as a single unit. For example, if any of <code><sql:update></code> tags inside the body of the <code><sql:transaction></code> tag fails, the entire transaction fails.
<code><sql:update></code>	JSTL defines an update as an SQL command that does not return a result set. SQL updates usually modify the database in some way, though this is not an absolute requirement. The changes made by the <code><sql:update></code> tag are immediate, unless the tag is being used as part of a <code><sql:transaction></code> block.

Table 8.8: Tags for interactions with databases [cf. Baye03, pp. 181-214]

In the example presented in listing 8.8, the tag library is used to retrieve user's identification number (`c_uid`), password (`c_pwd`), and access rights (`c_accesses`) from the `t_users` table. The collection of records is then displayed in a table. The database is accessed with the help of the ODBC bridge driver. The DNS name "myDB" had to be specified in the system resources (cf. lines 3-4 of the listing). The retrieved records are saved in the `users` variable and are subsequently presented in the form of an HTML table by looping through them with the `<c:forEach>` tag from the core library.

¹²¹ Connection pooling is a technique for sharing server resources among the requesting clients.

```

1. <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
2. <%@ taglib uri="http://java.sun.com/jstl/sql" prefix="sql" %>
3. <sql:setDataSource var="dataSource" driver="sun.jdbc.odbc.JdbcOdbcDriver"
4.   url="jdbc:odbc:myDB"/>
5. <html>
6.   <head>
7.     <title>Users</title>
8.   </head>
9.   <body>
10.    <sql:query var = "users" dataSource="{dataSource}">
11.      select c_uid,c_pwd,c_accesses from t_users
12.    </sql:query>
13.    <table border="1">
14.      <c:forEach var="row" items="{users.rows}">
15.        <tr>
16.          <td><c:out value="{row.c_uid}"/></td>
17.          <td><c:out value="{row.c_pwd}"/></td>
18.          <td><c:out value="{row.c_accesses}"/></td>
19.        </tr>
20.      </c:forEach>
21.    </table>
22.  </body>
23. </html>

```

Listing 8.8: Example of DBTags usage [based on Heat03, p. 167]

8.3.5 Wrappers

XML-based data sources can be easily integrated into adaptation frameworks since they clearly separate semantics from styling elements and allow for efficient transformations. HTML as a source of information is the most problematic format because presentation elements are mixed up with data. Special program routines, the so-called wrappers, have to be applied for automated extraction of information from Web sites and their conversion into a structured format (e.g. XML) [KuTr02; ChScSc02].

Wrappers should be able to download HTML pages from the Internet, recognize and extract predefined data and save them in a new format. Usually one wrapper is required for each individual Web site because of the varying structure of Web pages. Wrappers can be developed manually in a programming language using regular expressions or implemented with the help of specific toolkits that offer a GUI for faster development. Table 8.9 presents the most important features of chosen open-source wrappers that are able to convert HTML pages to XML format¹²². Their functionality is quite similar, some of them are command line-based, and some offer a more advanced GUI.

¹²² For a complete list of wrappers see [KuTr02] or <http://www.wifo.uni-mannheim.de/~kuhlins/wrappertools/index.html>.

Tool	Output Data	API	GUI	Editor	Script Language/ Regular Expressions	Short description
Araneus	XML, text	Yes	No	No	EDITOR	The generated wrapper is converted into Java code.
DEByE	XML, text, SQL database	No	Yes	No	No	Generation of wrappers with examples given by the user.
LAPIS	XML, text	Yes	Yes	Yes	Text Constraints	A tool with integrated browser, developed as a text editing tool.
Road Runner	XML, text	Yes	No	No	Regular expressions	Toolkit based on Araneus.
Web Language	XML, text	Yes	No	(Yes) Third Party Editor	WebL	Powerful toolkit that can cope with HTML forms, generate Web crawlers, and specify extraction rules.
XWrap	XML (incl. DTD)	Yes	Yes	No	No	Toolkit available only as a servlet that helps to generate and configure the wrapper. Generated wrappers are saved as Java source code, but do not always produce the expected result.

Table 8.9: Wrapper toolkits [KuTr02]

Web Language wrappers

In the MITL-based approach, content is retrieved from HTML pages with the help of wrappers programmed in Compaq's Web Language¹²³ [MaKi98]. This language was chosen because of its rich syntax, built-in support for common protocols (HTTP, FTP), integrated mechanisms for error handling, efficiency, and possibility to transform HTML documents into the XML format. Web Language is a high level, object-oriented scripting language that incorporates two novel features: service combinators and markup algebra. Service combinators are responsible for error handling. They are language constructs that provide reliable access to Internet services by emulating Web surfer's behavior when a failure occurs during a page retrieval. The markup algebra extracts structured and unstructured values from pages for computation, and is based on algebraic operations on sets of markup elements. It includes functions to extract elements and patterns from Web documents, operators to perform manipulations on extracted data, and functions to modify a page, for example by inserting or deleting its parts.

The processing model used in Web Language is depicted in figure 8.9. Web pages go through a pipeline of components. Retrieved sites (in HTML, XHTML, or XML) are parsed by a markup parser

¹²³ The old name of this wrapper was WebL.

and can be processed with the help of the markup algebra. The algebra facilitates the extraction of data from a page, performs computations on the retrieved values and manipulates the page. Subsequently, a new page is generated from its internal representation by the markup generator and the changed page can be saved.

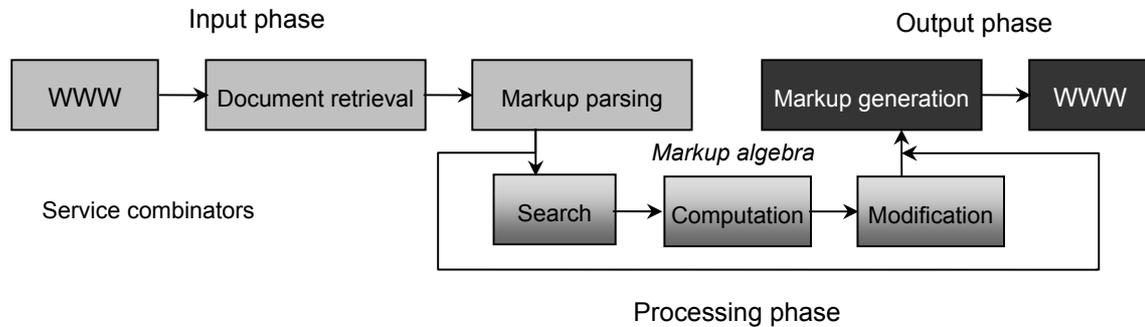


Figure 8.9: Processing model in Web Language [KiMa97, p. 3]

The Web Language includes data types, operators, statements, built-in functions and a number of standard modules that are displayed in table 8.10. It can be applied in Web shopping robots, meta-search engines, HTML analysis and checking routines, bots or business-to-business integration tools. In MITL the language is used to extract and manipulate information from HTML pages. The retrieved data is stored in XML files and it is subsequently processed. Web Language was implemented in Java; it is therefore possible to call Java objects directly from Web Language scripts or to code against its API instead of writing Web Language programs. PolyJSP and JRun support Web Language as scripting languages for JSP. Alternatively, there is a special Web Language servlet implemented from which one can execute the Web Language scripts [Mara99].

Module	Functionality
Base64	Encodes/decodes base 64 strings.
Browser	Provides access to a Web browser for displaying Web sites.
Cookies	Enables loading and saving of the HTTP cookie database.
Farm	Enables programming and controlling of several concurrently executing threads.
Files	Helps to process local files and download pages to files.
Java	Integration support for Java.
Servlet	Offers support for servlets.
Str	Functions for string manipulations.
Url	URLs manipulation functions.
WebCrawler	Extensible Web crawler object.
WebServer	Implementation of a simple Web server.

Table 8.10: Web Language modules [Mara99, pp. 113-114]

Listing 8.9 presents a simple example of a Web Language script. This program retrieves the `http://finance.yahoo.com` page, stores it into the variable “P” and checks all links on the page. The check is made with the help of a `Timeout` function that performs a function (`GetURL()`) and returns its value. If the evaluation takes more than the specified amount of time (in milliseconds), an exception is thrown. Therefore, if the program is not able to fetch a page with the `GetURL()` function, an appropriate message is displayed on the screen, the link is marked red and saved to a named HTML piece which contains the whole retrieved page. Subsequently, the modified page with dead links marked is displayed in a Web browser using the Browser’s module function `ShowPage()`.

```

1. import Browser;
2. var P = GetURL("http://finance.yahoo.com/");
3. every L in Elem(P, "a") do
4.   begin
5.     PrintLn("Checking ", L.href);
6.     Timeout(3000, GetURL(L.href))
7.   end;
8.   ?
9.   begin
10.    PrintLn(Text(L), " is dead");
11.    NewNamedPiece("font", Content(L)).color := "red";
12.   end;
13. end;
14. Browser_ShowPage(Markup(P));

```

Listing 8.9: Web Language example

8.3.6 Formats for Web feeds - RSS and Atom

RSS is a very popular standard and stands for Really Simple Syndication (RSS 2.0), Rich Site Summary (e.g. RSS 0.91, RSS 1.0), or RDF Site Summary (e.g. RSS 0.9 and 1.0), depending on the version [NC99; NWG05; RAB02; RWG00]. The format is a dialect of XML and is currently in version 2.0. RSS 2.0 is backwards compatible with old versions of the standard and is mainly used for the syndication of content and metadata over the Internet. RSS feeds enable users to receive newly released text or media content and integrate it into their Web pages.

The greatest improvement of RSS 2.0 in comparison to 1.0 is the ability to use own namespaces. The user can add new elements to the RSS feed that were not mentioned in the specification but for which a namespace was defined. An RSS file consists of a `<channel>` element and its subelements. A `<channel>` element consists of elements that represent metadata about the channel (`<title>`, `<link>`, and `<description>`) and of content in the form of items. Items can contain any content - an entry on a weblog, a complete article, a movie review, etc. Listing 8.10 presents an exemplary RSS 2.0 feed for Google jobs. The addition of the namespace declaration (`xmlns:g="http://base.google.com/ns/1.0"`) makes it possible to include five new attributes at the item level that were not specified in RSS 2.0 - `image_link`, `expiration_date`, `job_function`, `location`, and `label`.

```

1. <?xml version="1.0"?>
2. <rss version="2.0" xmlns:g="http://base.google.com/ns/1.0">
3. <channel>
4.   <title> Google Jobs</title>
5.   <link>http://www.google.com/support/jobs/</link>
6.   <description>Information about job openings at Google Inc.</description>
7.   <item>
8.     <title>HR Analyst - Mountain View </title>
9.     <link> http://www.google.com/jobs/topic.py?dep_id=1077&loc_id=1116</link>
10.    <description> We have an immediate need for an experienced analytical
        HR professional.The ideal candidate has a proven record
        of developing analytical frameworks to make fact-based
        decisions. </description>
11.    <g:image_link>http://www.google.com/images/google_sm.gif</g:image_link>
12.    <g:expiration_date>2005-11-15</g:expiration_date>
13.    <g:job_function>Analyst</g:job_function>
14.    <g:location>1600 Amphitheatre, Mountain View, CA, 94043, USA</g:location>
15.    <g:label>Hi-Tech</g:label>
16.    <g:label>Business development</g:label>
17.    <g:label>Personnel</g:label>
18.    <g:label>Silicon Valley</g:label>
19.    <g:label>Staffing</g:label>
20.  </item>
21. </channel>
22. </rss>

```

Listing 8.10: Exemplary RSS 2.0 feed for Google [http://www.google.com/base/help/rss_specs.html]

RSS 2.0	Atom
In RSS 2.0 plain text or escaped HTML can be used as a non-labeled payload.	Uses an explicitly labeled (i.e. typed) payload container.
The "description" element can contain a full entry or a summary.	Distinguishes between the "summary" and "content" elements. Allows the inclusion of non-textual content in the "content" element.
Uses its own "language" element.	Uses XML's built-in <code>xml:lang</code> attribute.
Does not possess means of differentiating between relative and non-relative URIs.	Uses the XML's built-in <code>xml:base</code> for relative URIs.
Is not standardized by any standardization body.	Is an open standard that has undergone the IETF standardization process.
No XML schema is used.	Includes an XML schema.
The RSS feeds are often sent as "application/rss+xml" although it is not a registered MIME type.	Has its own IANA-registered MIME-type.
Only full feed documents are supported.	Allows standalone Atom Entry documents.
The date format is not specified and different formats are used.	Dates have to be in the format described in RFC 3339.

Table 8.11: Basic differences between Atom 1.0 and RSS 2.0 [based on http://en.wikipedia.org/wiki/Atom_%28standard%29]

A new format for Web feeds is the Atom Syndication Format (Atom for short) specified in RFC 4287 and issued in December 2005 [NWG05]. The most relevant differences between Atom and RSS 2.0 are summarized in table 8.11. Listing 8.11 shows an exemplary Web feed in the Atom format.

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <feed xmlns="http://www.w3.org/2005/Atom">
3. <title>Atom Feed</title>
4. <subtitle>Some remarks</subtitle>
5. <link href="http://www.atomExamples.org/" />
6. <updated>2006-06-13T18:30:02Z</updated>
7. <author>
8.   <name>John Smith</name>
9.   <email>johnSmith@atomExamples.org</email>
10. </author>
11. <id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>
12. <entry>
13.   <title>Some title here </title>
14.   <link href="http://www.atomExamples/2005/12/13/atom03" />
15.   <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
16.   <updated>2006-06-13T18:30:02Z</updated>
17.   <summary>Some summary of the article here</summary>
18. </entry>
19. </feed>
```

Listing 8.11: Atom feed

Outline Processor Markup Language (OPML) is another popular format in content syndication - it is used for exchanging lists of RSS feeds between RSS aggregators [Wine05]. In order to process Web feeds different libraries, tools, and content readers/aggregators were developed. Some of the libraries and tools written in Java are introduced in table 8.12. Although ROME is currently the most comprehensive API for RSS and Atom processing, there is no good tag library based on this API. The best JSP tag library remains therefore RSS Utilities¹²⁴, developed in 2003.

¹²⁴ Cf. http://java.sun.com/developer/technicalArticles/javaserverpages/rss_utilities/.

Name of the library	Supported formats	Description/URL
Informa	RSS 0.9x, RSS 1.0, RDF, RSS 2.0, Atom 0.3, OPML	Informa is a comprehensive open-source Java framework that can be used for parsing, processing, and creating syndication feeds. http://informa.sourceforge.net
ROME	RSS (0.9x, RSS 1.0, RSS 2.0,) and Atom 0.3 and 1.0 feeds	The ROME API includes a set of parsers and generators for various types of syndication feeds. Additionally, it enables a conversion from one format to another. The most important drawback of this library is the lack of support for OPML. http://wiki.java.net/bin/view/Javawsxml/Rome
RSS JSP Tag Library	RSS 0.9x, RSS 1.0 / RDF, RSS 2.0, and Atom 0.3	The RSS JSP Tag library is supposed to provide an easy and flexible access to RSS and Atom feeds. The library is based on Informa. Unfortunately, no library can be downloaded from SourceForge so far. http://sourceforge.net/projects/rsstaglib/
RSS Utilities	RSS 0.91, 0.92 and 2.0	RSS Utilities is a JSP tag library to display RSS in JSPs. It includes its own parser and is independent from any other RSS APIs. http://java.sun.com/developer/technicalArticles/javaserverpages/rss_utilities/
RSS4J	RSS 0.9x, 1.0, and 2.0	RSSLib4J is a Java API for parsing and retrieving information from RSS feeds. http://sourceforge.net/projects/rsslib4j/
RSS-desk	RSS 0.9x, RSS 1.0 / RDF, RSS 2.0, and Atom 0.3	RSS-desk is a reusable component for news aggregation, based on the Informa API. It includes a JSP tag library with limited capabilities. http://sourceforge.net/projects/rss-desk/
RSSLibJ	RSS 0.92	RSSLibJ is a Java class library for the generation of RSS data in various formats, based on a simple object model. http://enigmastation.com/rsslibj/
Sandler	Atom, XML formats	Sandler is an Atom parser and manipulation library implemented in Java. http://sandler.sourceforge.net

Table 8.12: Java-based RSS/Atom libraries

8.4 MITL tags

The developed library for device-independent content delivery to mobile devices is a server-side approach, especially useful for designing pages with medium complexity. The Mobile Interfaces Tag Library (MITL) generates appropriate markup elements in WML, XHTML, HTML, and Java ME (only high-level UIs) depending on the device context. It takes care of the pagination and creation of navigation elements and was designed to avoid the high complexity level of UIDLs such as UIML or RIML. Furthermore the library was developed taking into account the requirements for device-independent content delivery specified by the W3C Consortium. MITL is easy to use and enables rapid prototyping of applications designed for different devices. Authors have to include only one tag for displaying a particular element in four different formats.

The defined tags offer a Web application developer the possibility to control how data is processed in back-end Java components without including any Java code in the JSP page. The tags allow

furthermore for the generation of completely different presentation layers. Content transformed by MITL can be in the form of regular text, HTML pages, data extracted from a database, or in the XML format. MITL can be used with other JSP tag libraries, for example with the JSTL tag library introduced in the previous section. It is also possible to combine MITL tags with any Java code or markup-specific tags.

The request processing performed with the help of MITL encompasses the following steps: A client sends a request for a JSP page and delivers information about device capabilities in the form of HTTP headers or CC/PP profiles. A Web server (e.g. Apache Tomcat) looks for a document that was requested and processes it on the server side. MITL retrieves information about the device context (supported formats, screen size, etc.) using the data obtained from HTTP headers or extracting relevant information from a local or external repository of CC/PP profiles. Then it converts each tag in the document to an appropriate element in a particular format or generates a Java ME application from these tags. For example the `<mitl:doc title="Welcome">` tag is converted into the code: `<HTML> <HEAD> <TITLE> Welcome </TITLE> </HEAD> <BODY> [...] </BODY> </HTML>` for browsers supporting HTML. Information about Java classes responsible for the conversion is stored in the XML format in tag library descriptor files (TLD). The output is sent back as a response to the client and is displayed in a browser. Alternatively, the user obtains a link to the Java ME application and can download a MIDlet. This process is illustrated in figure 8.10.

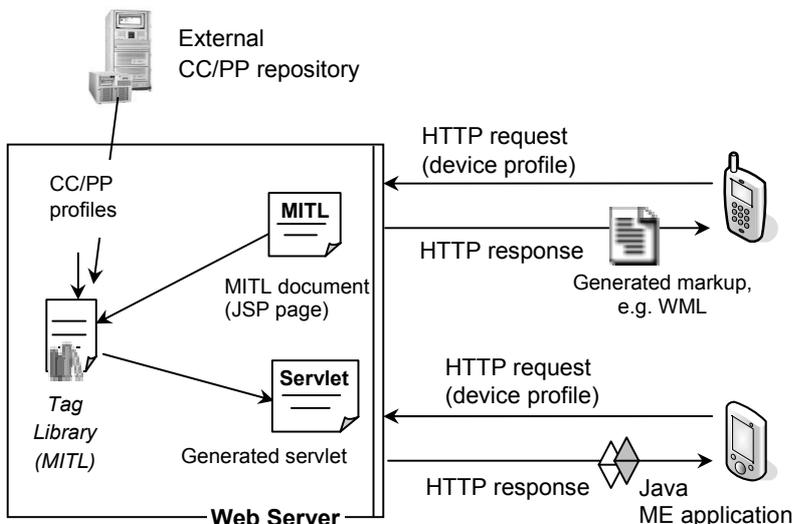


Figure 8.10: Request processing with MITL

MITL contains the following tags (mandatory attributes are marked in bold type):

- **DocTag**

```

syntax: <mitl:doc title="" stylesheet="" altstylesheet=""
        defaultCSS="silver/green/blue/orange/brown" preferredApp="[array]">

```

Depending on the detected browser type, the `DocTag` generates the root elements for HTML, WML, or XHTML and the title of a page, as specified in the `title` attribute. If the device supports Java ME applications, the `DocTag` produces a skeleton of a MIDlet. The `title` attribute is then used to

generate the name of the MIDlet and the name of the package, to which the MIDlet belongs. The tag can provide information about a link to an appropriate CSS style sheet in XHTML and HTML (the `stylesheet` attribute). Alternatively, the `DocTag` is able to automatically generate a style sheet, basing on the retrieved device characteristics. The created style sheet is saved under a name specified in the `altstylesheet` attribute.

Since responses containing a markup are sent by a server, the tag is in charge of setting a correct MIME type for each device (e.g. "text/html" for HTML, "text/xml+xhtml" for XHTML, etc.). The developer can also benefit from one of the five default style sheets (silver, green, blue, orange, and brown) that specify the formatting of an HTML page or, alternatively, modify these style sheets. Additionally, the preferred hierarchy of markup languages and application types can be specified in the `preferredApp` attribute. If the device supports, for example, XHTML and Java ME and the user prefers Java ME over XHTML than a MIDlet is generated. The preferences can be provided by the user in each request, saved in a database and retrieved from it or the hierarchy of preferences can be explicitly set by a developer.

The `DocTag` is always the first tag in a page and it retrieves the information about device features. Tags are able to communicate with other tags used on the same page (e.g. the `DocTag` may be called from another tag by using the following syntax: `DocTag parent=(DocTag) findAncestorWithClass(this, DocTag.class);`). The `DocTag` detects, for example, the browser type and its capabilities and all other tags within a page can use this information for content adaptation. It is not necessary for each tag to query a device for its capabilities. This decreases the number of data exchanged between a client and a server and enables faster processing of requests.

The `DocTag` determines the form of presentation depending on the relevant features of a device. It uses the CC/PP profiles to obtain the delivery context, or, if no CC/PP is provided, the HTTP headers. Relevant features that can be obtained from a CC/PP profile are as follows¹²⁵:

- supported markup languages or, alternatively, support for Java ME
(`myProfile.getAttribute("CcppAccept");`)
- HTML or XHTML version supported by the browser (`myProfile.getAttribute("HtmlVersion"/"XhtmlVersion");`)
- support for tables (`myProfile.getAttribute("TablesCapable");`)
- screen size, its width and height (`myProfile.getAttribute("ScreenSize");`)
- character size (`myProfile.getAttribute("ScreenSizeChar");`)
- support for colors (`myProfile.getAttribute("ColorCapable");`)
- deck size (`myProfile.getAttribute("WmlDeckSize");`)
- supported image types, e.g. JPG, BMP, WBMP, PNG, GIF, no support
(`myProfile.getAttribute("CcppAccept");`)

¹²⁵ For each feature, the appropriate DELI method that can be used to retrieve the feature is also provided.

- number of supported colors (`myProfile.getAttribute("BitsPerPixel");`)
- number of softkeys (`myProfile.getAttribute("NumberOfSoftKeys");`)
- proportionality of the standard font used by a mobile device (`myProfile.getAttribute("StandardFontProportional");`)
- ratio of the width of one pixel to the height of one pixel (`myProfile.getAttribute("PixelAspectRatio");`)
- character sets used for text input (`myProfile.getAttribute("InputCharSet");`)
- capability to run Java programs and the supported version of Java platform (`myProfile.getAttribute("JavaEnabled"/"JavaPlatform");`)
- name, vendor, and version number of the operating system of a wireless device (`myProfile.getAttribute("OSName"/"OSVendor"/"OSVersion");`)
- name and version of the browser (`myProfile.getAttribute("BrowserName"/"BrowserVersion");`).

The information retrieved from HTTP headers is less comprehensive and includes:

- supported markup language (alternatively Java ME) together with the media type quality factors and supported image types (the "Accept" attribute)
- acceptable character sets and languages ("Accept-Charset", "Accept-Language")
- screen size in pixels ("X-UP-DEVCAP-SCREENPIXELS")
- supported colors ("X-UP-DEVCAP-ISCOLOR")
- browser type, device manufacturers, device version number, device hardware, and used browsers ("User-Agent")
- deck size ("X-UP-DEVCAP-MAX-PDU")
- number of soft keys ("X-UP-DEVCAP-NUMSOFTKEYS")
- URL to the UAPProfile ("X-WAP-PROFILE").

Listing 8.12 presents the most important excerpts from the implementation of the `DocTag`. After the declaration of a package (line 1) and necessary imports (lines 2-12), the variables that are used further in this tag are declared (lines 14-33). Subsequently, the setter and getter methods are implemented to guarantee access to relevant variables with a private access (to which usually the retrieved device properties are assigned) from related tags (cf. lines 34-47). Since the number of properties is quite large, the listing presents only some exemplary methods for setting and getting the values of variables.

The `doStartTag()` method encapsulates the business logic of the `DocTag` and produces different markups or Java ME code (cf. lines 105-142), depending on the features of the device. Delivery context is retrieved from CC/PP profiles (saved in XML files) with the help of DELI library (cf. lines 55-88) or from HTTP headers (cf. lines 90-103). Some properties cannot be obtained directly and have to be computed. For example, in order to get the maximum font size supported by a device the height of a screen has to be divided by the height of the screen expressed in characters using the maximum font size. Basing on the obtained device features, an appropriate code is created. For browsers supporting HTML and XHTML, it is possible to generate automatically a style sheet using the method `generateCSS(nameCSS, screenWidth, maxFont, color, tablesCapable)` from the `CSSGenerator` helper class. The parameters passed to this method (the proposed name for the CSS

file, screen size, maximal font size, support for colors, the capability to support tables) influence the content of the CSS file. Design principles for the generation of style sheets designated for handhelds also have impact on the content of the CSS file. For example, a CSS file generated by the method `generateCSS(myStylesheet, 208, 33, 1, 0)` for Nokia 7210 device (cf. Appendix 4 for the profile) produces a style sheet presented in listing 8.13. Important design rules for mobile CSS are summarized in table 8.13.

Features	Implementation
Body properties	Do not use pixels for formatting elements that are larger than 5px. Use ems or percentages instead. Reduce margins, padding, and border widths to suit small screens. Fixed-size elements include images and form controls. Assign them a maximum width of 100%.
Images	Make images fit window size.
Navigation labels, headers	Use text rather than images for navigation labels and headers, keep textual descriptions as short as possible.
Hidden elements	Use the “ <code>display:none</code> ” property to hide content on mobile devices.
Text blocks	Keep text blocks as wide as possible by using only small amounts of horizontal spacing (1em if possible). Specifying a margin in percentages will also work well on a wide variety of devices.
Layout	Collapse the layout into one column by transforming the cells into blocks. Override the rules that add floating and absolute positioning behavior.
Overflow	Avoid overflow in the horizontal direction on small screens. Make sure wide elements fit the narrow screen.
Font size, preformatted text	Limit text size. Default sizing of text should be consistent with individual devices' standard font sizes. A large font type should not be larger than twice the size of the font used for text in paragraphs. For preformatted blocks (<code><pre></code>), either wrap the text very tightly (25-30 characters) or allow long lines to break.

Table 8.13: Design principles for mobile CSS [based on Moll05; Wilk04]

For devices supporting Java ME, a skeleton of a MIDlet (as displayed in listing 8.14) and the templates for a manifest and JAD file are generated with the help of the method `generateSkeletonJAD(title)` from the `J2MEGenerator` helper class. Further tags add appropriate elements to this code and insert them in right places. Finally, the `doEndTag()` method generates the closing tags for the selected markup language (lines 153-163). For a MIDlet, it compiles the available Java files, builds a JAR and a JAD file and produces a page with a link to the JAR (`produceJARJAD(title)`, `generateOTA(title)` methods from the `J2MEGenerator` class, respectively). The `log()` methods are used for logging exceptions (cf. lines 176-179).

The `analyzePrefCap(String acceptHTTP, String [] preferredApp, String [] supportedApp)` method helps to retrieve the end format that should be delivered to the device. If a device supports many formats, the method takes into consideration the supported markups and these preferred by the user. If the user prefers to obtain XHTML over WML and the device supports both

formats, XHTML will be sent as a response. If the user did not set any preferences, and many formats are supported, Java ME will be delivered first followed by XHTML, WML, and HTML (cf. lines 182-205).

The `MobileYes/MobileNoTag` and the `WMLDeckTag` are device-specific and influence only the output for mobile devices equipped with a browser.

- **MobileYes/MobileNoTag**

syntax: `<mitl:mobileYes>`, `<mitl:mobileNo>`

The `MobileYesTag/MobileNoTag` were developed to include/exclude page fragments in/from mobile pages written in XHTML, WML, or HTML. The `MobileYesTag` indicates that elements enclosed in these tags should be included in the output for mobile browsers supporting WML, XHTML, or HTML. The `MobileNoTag` enables the elements enclosed by these tags to be excluded in the output for mobile browsers and presented only in desktop browsers with XHTML or HTML support and in Java ME applications. There is no need to use the `MobileYesTag`, if no `MobileNoTag` was previously applied, because by default all four input types can be generated and its delivery is device-dependent.

- **DeckTag**

syntax: `<mitl:deck title="" cardID="" timer="" task="" nextCard="">`

It produces a WML card or a deck of cards. For XHTML and Java ME applications the `DeckTag` is used to divide a page into sub-pages. For each card a card's title and a name is generated. It also offers the possibility to specify tasks and "onenter" events (such as "onenterforward" or "accept", etc.) or the amount of time after which the user will be redirected to a new page. For example, the following tag:

```
<mitl:deck title="Welcome" cardID="card1" task="accept/prev/next/pick"
nextCard="card2"> [...]</mitl:deck>
```

produces the code:

```
<card id="card1" title="Welcome" newcontext="true">
  <do type="accept" label="Next">
    <go href="card2"/>
  </do>
  <p>[...]</p>
</card>
```

```
1. package uni.euv.bj.mitl;

2. import javax.servlet.http.*;
3. import javax.servlet.*;
4. import javax.servlet.jsp.*;
5. import java.net.*;
6. import javax.servlet.jsp.tagext.*;
7. import java.io.*;
8. import java.util.*;
9. import com.hp.hpl.deli.*;
10. import javax.xml.parsers.*;
11. import uni.euv.bj.mitl.CSSGenerator;
12. import uni.euv.bj.mitl.J2MEGenerator;

13. public class DocTag extends TagSupport {
14.     private String title, javaPlatform, osName, osVendor, osVersion;
15.     private String stylesheet, altstylesheet, defaultCSS;
16.     private int markup, tablesCapable, imageType;
17.     private int screenW, screenH, maxFont;
18.     private int sCharWidth, sCharHeight, noSoftKEys, noColors, propFont;
19.     private String xhtmlVersion, htmlVersion, browserName, browserVersion;
20.     private int color, javaEnabled, pixelW, pixelH;
21.     private String charset, lang, UAProfLink;
22.     private JspWriter out;
23.     private ServletContext application;
24.     private HttpServletRequest request;
25.     private HttpServletResponse response;
26.     private ServletConfig config;
27.     private Profile myProfile;
28.     String [] arrImg={"txt","jpeg","png","gif","x-xbitmap","wbmp"};
29.     private String [] preferredApp;
30.     String [] supportedApp={"xhtml","j2me","wml","html"};
31.     private String title, deckSize;
32.     int flag=1;
33.     int i,j,k;

34.     public void setTitle(String title){
35.         this.title = title; }
36.     public void setStylesheet(String stylesheet){
37.         this.stylesheet = stylesheet;}
38.     public void setPreferredApp(String [] preferredApp){
39.         this.preferredApp = preferredApp;}
40.     public void setMarkup(int markup){
41.         this.markup=markup;}
42.     public int getMarkup(){
43.         return markup;}

44.     public void setScreenWidth(int screenWidth){
45.         this.screenWidth=screenWidth;}
46.     public int getScreenWidth(){
47.         return screenWidth;}
```

[further set and get methods]

```

48. public int doStartTag() throws JspException {
49.     try {
50.         request = (HttpServletRequest) pageContext.getRequest();
51.         response = (HttpServletResponse) pageContext.getResponse();
52.         out = pageContext.getOut();
53.         out.clear();
54.         application = pageContext.getServletContext();
55.         Workspace.getInstance().configure(application, "config/deliConfig.xml");
56.         myProfile = new Profile(request);

// reading attributes from CC/PP profiles

57.         if (myProfile.size()>0){
58.             if (myProfile.getAttribute("CcppAccept")!=null){
59.                 String acceptCCPP =
60.                 myProfile.getAttribute("CcppAccept").getValue().toString();
61.                 setMarkup(analyzePrefCap(acceptCCPP,
62.                     supportedApp, getPreferredApp()));
63.                 markup = getMarkup();
64.                 while ((flag==1) && (i<6)){
65.                     if (acceptCCPP.indexOf(arrImg[i]) !=-1) {
66.                         setImage(i);
67.                         flag=0; }
68.                         i=i+1;}
69.                 if (flag==1){
70.                     setImage(i);
71.                     flag=0;}}

72.                 if (myProfile.getAttribute("ScreenSize")!=null){
73.                     String screenSizeTemp =
74.                     myProfile.getAttribute("ScreenSize").getValue().toString();
75.                     int sLength = screenSizeTemp.length();
76.                     int indexX = screenSizeTemp.indexOf("x");
77.                     screenH = Integer.parseInt
78.                     (screenSizeTemp.substring(1, indexX));
79.                     screenW = Integer.parseInt
80.                     (screenSizeTemp.substring(indexX+1, sLength-1));
81.                     setScreenHeight(screenH);
82.                     setScreenWidth(screenW);}

83.                 if (myProfile.getAttribute("ScreenSizeChar")!=null){
84.                     String sSizeCharTemp =
85.                     myProfile.getAttribute("ScreenSizeChar").getValue().toString();
86.                     int sLength1 = sSizeCharTemp.length();
87.                     int indexX1 = sSizeCharTemp.indexOf("x");
88.                     String height1 = sSizeCharTemp.substring(1, indexX1);
89.                     String width1 =
90.                     sSizeCharTemp.substring(indexX1+1, sLength1-1);
91.                     setSCharHeight(Integer.parseInt(height1));
92.                     setSCharWidth(Integer.parseInt(width1));}
93.             }

```

```

[reading and setting further properties from CC/PP]

89.         else {

// retrieving device properties from HTTP headers

90.         String acceptHTTP =request.getHeader ("Accept");
91.         setMarkup(analyzePrefCap(acceptHTTP,
92.             supportedApp,getPrefferedApp()));
           markup = getMarkup();

// Method to get the maximum packet size supported by a device

93.         deckSize = request.getHeader("X-UP-DEVCAP-MAX-PDU");
94.         if (deckSize!=null){
95.             setDeckSize(Integer.parseInt(deckSize)); }

// Method to get the display area of the device

96.         String displ = request.getHeader("X-UP-DEVCAP-SCREENPIXELS");
97.         if (display!=null){
98.             int sLength = displ.length();
99.             int indexX = displ.indexOf(",");
100.            screenH = Integer.parseInt(displ.substring(0, indexX));
101.            screenW =
102.                Integer.parseInt(displ.substring(indexX+1,sLength));
103.            setScreenHeight(screenH);
104.            setScreenWidth(screenW);}

[reading and setting further properties from HTTP headers]
104.     }
// generating output

105.     switch (markup){

// HTML
106.         case 1:
107.             response.setContentType("text/html");
108.             out.println("<html>");
109.             out.println("<head><title>"+title+"</title>");
110.             if (altStylesheet != null){
111.                 CSSGenerator gHTML= new CSSGenerator();
112.                 gHTML.generateCSS(altStylesheet, getScreenWidth(),
113.                     getMaxFont(), getColor(), getTablesCapable());
114.                 out.println("<link rel=\"stylesheet\" href=\""+altStylesheet+"\"
115.                     type=\"text/css\"/>");}
116.             else if (stylesheet != null){
117.                 out.println("<link rel=\"stylesheet\" href=\""+stylesheet+"\"
118.                     type=\"text/css\"/>");}
119.             else if (defaultCSS != null){
120.                 out.println("<link rel=\"stylesheet\" href=\""+defaultCSS+"\"
121.                     type=\"text/css\"/>");}
122.             out.println("</head><body>");
123.             break;

```

```

// WML

120.     case 2:
121.         response.setContentType("text/vnd.wap.wml");
122.         out.println("<?xml version=\"1.0\"?>");
123.         out.println("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.3//EN\"
            \"http://www.wapforum.org/DTD/wml13.dtd\">");
124.         out.println("<wml>");
125.         out.println("<template>");
126.         out.println("<do type=\"options\" label=\"Back\">");
127.         out.println("<prev />");
128.         out.println("</do>");
129.         out.println("</template>");
130.     break;

// XHTML

131.     case 3:
132.         response.setContentType("text/html");
133.         out.println("<?xml version=\"1.0\"?>");
134.         out.println("<!DOCTYPE html PUBLIC \"-//WAPFORUM//DTD XHTML Mobile
            1.0//EN\" \"http://www.wapforum.org/DTD/xhtml-mobile10.dtd\">");
135.         out.println("<html xmlns=\"http://www.w3.org/1999/xhtml\">");
136.         out.println("<head><title>"+title+"</title>");
137.         if (altStylesheet != null){

            [generating a CSS and the rest of XHTML, similarly to HTML]

138.         break;
            }
//J2ME

139.     case 4:
140.         J2MEGenerator myGenerator = new J2MEGenerator();
141.         myGenerator.generateSkeleton(getTitle());
142.     break; }

//exception handling

143.     catch(IOException ioe) {
144.         log("Error in the tag: " + ioe); }
145.     catch (ParserConfigurationException xmlP){
146.         log("Error in the tag: " + xmlP); }
147.     catch(Exception e) {
148.         log("An error occurred");
149.         throw new JspTagException("Error in the tag!" + e);}
150.     return EVAL_BODY_INCLUDE;
151. }

```

```

152. public int doEndTag() throws JspException {
153.     try {
154.         switch (markup) {
155.             case 1:
156.                 out.println("</body></html>");
157.                 break;
158.             case 2:
159.                 out.println("</wml>");
160.                 break;
161.             case 3:
162.                 out.println("</body></html>");
163.                 break;
164.             case 4:
165.                 J2MEGenerator myEndGenerator = new J2MEGenerator();
166.                 myEndGenerator.produceJARJAD(getTitle());
167.                 myEndGenerator.generateOTA(getTitle());
168.                 break;}}
169.     catch(IOException ioe) {
170.         log ("Error in the tag: " + ioe);}
171.     catch(Exception e) {
172.         log("An error occurred");
173.         throw new JspTagException("Error in the tag!" + e);}
174.     return EVAL_PAGE; }
175.     public void release(){}
176.     protected void log(String msg) {
177.         getServletContext().log(msg); }
178.     protected void log(String msg, Throwable t) {
179.         getServletContext().log(msg, t); }
180.     protected ServletContext getServletContext(){
181.         return pageContext.getServletContext();}

182.     public static int analyzePrefCap(String acceptHTTP, String []
183.         preferredApp, String [] supportedApp){
184.         String accept="";
185.         int j,i,k=0;
186.         String [] supportedDevApp;
187.         while ((j<4)){
188.             if (acceptHTTP.indexOf(supportedApp[j]) !=-1) {
189.                 supportedDevApp[j]= supportedApp[j];}
190.             }
191.             j=j+1;}
192.         if (supportedDevApp.length>0){
193.             while (k<preferredApp.length){
194.                 if(preferredApp[k].equals(supportedDevApp[k])){
195.                     accept = preferredApp[k];
196.                     break;}
197.                 k=k+1;}
198.             if accept.equals("") accept=acceptHTTP;
199.             if ((accept.indexOf("j2me") !=-1) || (accept.indexOf("java") !=-1)){
200.                 markup=4;}
201.             else if (accept.indexOf("xhtml") !=-1) {
202.                 markup=3;}
203.             else if (accept.indexOf("wml") !=-1) {
204.                 markup= 2;}
205.             else {markup= 1;}
206.             return markup;} }

```

Listing 8.12: Relevant fragments of DocTag

```

1. body {
2.     max-width: 206px ! important;
3.     padding: 1px ! important;
4.     margin: 0 ! important;
5.     white-space: normal ! important;
6.     font-family: Tahoma, Verdana, Arial, Sans-Serif; }
7. * {
8.     font-size: 12px ! important;
9.     padding: 1px ! important;
10.    text-align: left ! important;
11.    line-height: 1.01 em ! important; }
12. table, tbody, thead, tfoot, tr, td, th, col, colgroup {
13.    display: block ! important; }
14. img {
15.    max-width: 200px ! important; }
16. img[width="1"], img[height="1"], img[width="468"], img[height="600"] {
17.    display: none ! important; }
18. ul li {
19.    float: none ! important; }
20. li {
21.    list-style-position: inside ! important;
22.    display: inline !important; }
23. #menu ul li a {
24.    width: auto;
25.    height: 1.2em;
26.    border: 1px solid #ccc;
27.    margin: 1px 0;
28.    line-height: 1.2em; }
29. input, textarea, select {
30.    max-width: 100%; }
31. #formarea label {
32.    font-size: 0.90em;
33.    margin-top: 0.4em; }
34. .postdata {
35.    font-size: 0.83em;
36.    margin: 0.83em 0 0.4em 0;
37.    border: 0;
38.    border-bottom: 1px dotted #ccc;
39.    border-top: 1px solid #f3f3f3 }
40. .postbody {
41.    margin: 0 0 1.6em 0;
42.    line-height: 1.4; }
43. div {
44.    padding: 0; margin: 0 ! important; }
45. p {
46.    text-align: justify; padding: 0; margin: 5px 0 }
47. a[href] {
48.    text-decoration: underline ! important;
49.    display: inline-block ! important; }
50. pre {
51.    white-space: pre-wrap ! important; }

```

Listing 8.13: CSS file generated automatically for Nokia 7210

```
1. package helloworld;

2. import javax.microedition.midlet.*;
3. import javax.microedition.lcdui.*;

4. public class HelloWorld extends MIDlet {
5.     public HelloWorld () {
6.         initialize();
7.     }

8.     private void initialize() {
9.         // Pre-init code should be inserted here at runtime
10.        // Post-init code should be inserted here at runtime
11.    }

12. public Display getDisplay() {
13.     return Display.getDisplay(this);
14. }

15. public void exitMIDlet() {
16.     getDisplay().setCurrent(null);
17.     destroyApp(true);
18. }

19. public void startApp() {}

20. public void pauseApp() {}

21. public void destroyApp(boolean unconditional) {}
22. }
```

Listing 8.14: Java ME skeleton

- **ParagraphTag**

syntax: <mitl:paragraph cssStyle="">

The ParagraphTag produces a paragraph (<p>) in (X)HTML and WML. The alternative attribute `cssStyle` specifies the formatting of a paragraph in (X)HTML (e.g. font size, color, etc.).

- **BoldTag, ItalicTag**

syntax: <mitl:bold>, <mitl:italic>

The BoldTag formats text to be bolded, whereas the ItalicTag makes it italic. In Java ME applications, both tags can be applied to StringItems by setting the font constant to `FONT.BOLD` or `FONT.ITALIC`. If both tags, the BoldTag and the ItalicTag, were applied the code looks like that:

```
Font_1 = Font.getFont(Font.FACE_MONOSPACE,
Font.STYLE_BOLD+Font.STYLE_ITALIC, Font.SIZE_MEDIUM));
Text_1.setFont(Font_1);
```

- **HeadingTag**

syntax: <mitl:heading level="1-6" align="left/right/center">

The `HeadingTag` produces headings (<h>) in XHTML and HTML. The `level` attribute (from 1 to 6) specifies the font size for a heading (e.g. <h1> corresponds to 24 points and <h6> to 8 points) . For WML the `HeadingTag` with levels 1 to 3 generates the and <u> tags (bold and underlined). The `align` attribute indicates the alignment of the heading (left, right, center).

- **BreakTag**

syntax: <mitl:break height="" width=""/>

The `BreakTag` produces a break (
) in XHTML, HTML, and WML. In Java ME, a `Spacer` component [Goya05] is generated. The `Spacer` is used in order to position UI elements and puts some space between other components. The element is invisible and its size is determined by the attributes `height` and `width`.

- **PaginationTag, ItemsTag**

syntax: <mitl:pagination url="" midpElID="" cssStyle="">, <mitl:items>

The `PaginationTag` is responsible for the pagination of larger data sets and emulates the “previous item/next item” style. The `url` attribute is used as a basis URL for the generated links – by default the URL of the current page is taken to produce further pages and appropriate links to them. The `cssStyle` attribute allows to define the CSS formatting for the “previous/next” links. The `ItemsTag` specifies the data set that will be paginated.

For a MIDlet, the index is not displayed at the bottom of the page but a separate navigation menu is built. This menu is based on objects from the `Command` class. This class encapsulates the semantic information about an action. The action itself is defined in a `CommandListener` object that is associated with a screen. A `Command` constructor takes three parameters: a label, a type, and a priority, e.g. `nextCommand = new Command("Next", Command.SCREEN, 2)`. The label parameter is used for the visual representation of the command, the type specifies its purpose, and the priority indicates the priority of this command relatively to other screen commands. The higher the priority of a command, the lower its importance. The defined command types are specified in table 8.14.

The `PaginationTag` produces 2 `Command` objects with the labels “Previous” and “Next”, adds them to a form, and generates appropriate `CommandListeners`. When the user selects the “Next” command, he/she is redirected to a new form or UI element with a name built from the `midpElID` attribute. The result of the `PaginationTag` for a Java ME application is displayed in figure 8.11.

Command Type Constant	Description
<code>public static int BACK</code>	Return to the logically previous screen.
<code>public static int CANCEL</code>	A standard negative answer to a dialog query.
<code>public static int EXIT</code>	Indication to exit the application.
<code>public static int HELP</code>	A request for online help.
<code>public static int ITEM</code>	A hint for the application that the command is related to a specific item on the screen (e.g. the selected item).
<code>public static int OK</code>	A standard positive answer to a dialog query.
<code>public static int SCREEN</code>	An application-defined command related to the currently displayed screen.
<code>public static int STOP</code>	Stop a currently executing operation.

Table 8.14: Command type constants [Piro02, p. 50]

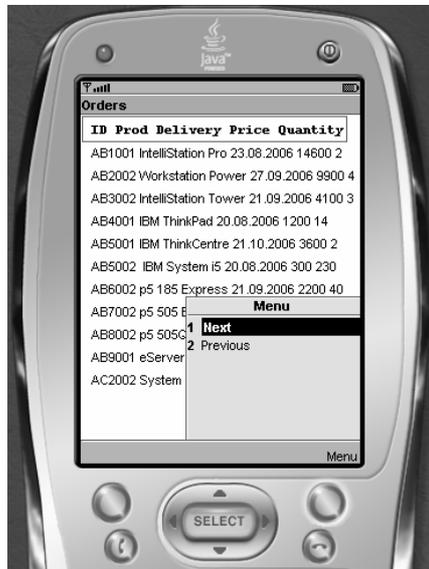


Figure 8.11: Results produced by *PaginationTag* (for a MIDlet)

The *PaginationTag* was kept simple since there already exists an advanced tag library for data pagination offering a lot of options - the *PagerTag Library*¹²⁶. However, for an inexperienced programmer the library may be difficult to use. The *PagerTag* library consists of a set of tags and offers the possibility to define the data sets, to create a navigation index and its elements (e.g. first item, next item, items that should be skip, etc.). Different navigation styles, known from the Web such as Altavista's or Google's style, may be applied to the index.

Table 8.15 presents the most important tags with their attributes and provides a short description of them. Listing 8.15 shows an example of using the *PagerTag* library for the generation of an index in the Altavista's style. After the specification of the URL used and the maximum number of pages in the

¹²⁶ Cf. <http://jsptags.com/tags/navigation/pager/pager-taglib-2.0.html>.

index (20), the request parameters and the items that should be paginated are specified (lines 6-10). Subsequently, the links for the previous and next pages are defined with the `<pg:prev>`, `<pg:next>` tags and links to each item in the index are generated with the `<pg:pages>` tag. If the number in the index is smaller than 10, an additional blank is produced, if the number matches the current page it is displayed without a link and in bold. Otherwise a number with a link to appropriate page is created. The generated result is displayed in figure 8.12.

Tags	Description
<code><pg: pager id="value" url="url" index="center forward half-full" items="int" isOffset="true false" maxItems="int" maxPageItems="int" maxIndexPages="int" export="expression" scope="page request"></code>	Defines the pagination. The <code>url</code> attribute specifies the URL that will be used for the index. The <code>index</code> attribute defines the behavior of the index, the <code>items</code> attribute specifies the number of items in the data set, the <code>maxPageItems</code> attribute specifies the maximum number of items per page, the <code>export</code> attribute defines the variables to export and the <code>scope</code> attribute specifies their scopes.
<code><pg:param id="value" name="value" value="value"></code>	Adds request parameters to each page URL in the index.
<code><pg: item id="value"></code>	Defines items of information.
<code><pg: index id="value" export="expression"></code>	Defines a navigation index.
<code><pg: first id="value" unless="current indexed" export="expression"></code>	Creates a link to the first page. The <code>unless</code> attribute specifies when the first page link should not be displayed.
<code><pg: prev id="value" ifnull="true false" export="expression"></code>	Creates a link to the previous page. The <code>ifnull</code> attribute specifies whether the body should be evaluated if there is no previous page.
<code><pg: page id="value" export="expression"></code>	Creates a link to the current page or exports information about the current page.
<code><pg:pages id="value" export="expression"></code>	Creates a link to each page in the index.
<code><pg:next id="value" ifnull="true false" export="expression"></code>	Creates a link to the next page.
<code><pg:last id="value" unless="current indexed" export="expression"></code>	Creates a link to the last page.
<code><pg:skip id="value" ifnull="true false" pages="int" export="expression"></code>	Creates a link to the page number calculated by adding the value of the <code>pages</code> attribute to the current page number.

Table 8.15: Tags and attributes in PagerTag library [<http://jsptags.com/tags/navigation/pager/pager-taglib-2.0.html>]

```

1. <%taglib uri="http://jsptags.com/tags/navigation/pager" prefix="pg" %>
2. <pg:pager url="http://www.altavista.com/cgi-bin/query" maxIndexPages="20"
3.     export="currentPageNumber=pageNumber">
4.     <pg:param name="pg"/>
5.     <pg:param name="q"/>
6.     <ex:searchresults>
7.         <pg:item>
8.             < %=searchResult %>
9.         </pg:item>
10.    </ex:searchresults>
11.    <pg:index> Result pages:
12.        <pg:prev>&nbsp;<a href="<%=pageUrl%>">[&lt;&lt; Prev]</a></pg:prev>
13.        <pg:pages><%
14.            if (pageNumber.intValue()<10){
15.                %> &nbsp;<%
16.            }
17.            if(pageNumber == currentPageNumber){
18.                %><b><% pageNumber %></b><%
19.            else {
20.                %><a href="<%=pageUrl %>"><%= pageNumber %></a><%
21.            } %>
22.        </pg: pages>
23.        <pg:next>&nbsp;<a href="<%=pageUrl%>"> [Next &gt;&gt;]</a></pg:next>
24.        </br>
25.    </pg:index>
26. </pg:pager>

```

Listing 8.15: Generation of Altavista index with PagerTag library

Result Pages: [[<< Prev](#)] [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#) [[Next >>](#)]

Figure 8.12: Index generated by PagerTag library

- **FragmntOnTag, TitleTag, DescTag, DetailsTag**

syntax:

```

<mitl:FragmntOn>
<mitl:title midpElID="" cssStyle="">
<mitl:desc midpElID="" cssStyle="">
<mitl:details midpElID="" cssStyle="">

```

These tags are used for automatic fragmentation of text according to the screen size. The `FragmntOnTag` identifies these text fragments that should be taken into account in the fragmentation process. The `TitleTag`, `DescTag`, and `DetailsTag` categorize the basic text elements such as title, general description, and details. According to this classification, text is displayed on a screen of a mobile device. The title has the highest priority and is displayed as the first item on the page, followed

by a description and eventually some details. On very small screens only the title is presented and it becomes a link to further details. If the descriptions or details are extremely long, they can be further split and shown on more screens. The splitting mechanism takes into account the average number of characters that can be displayed on one screen (per row and column) and uses an advanced `StringTokenizer` class (based on the Sun's `StringTokenizer` class¹²⁷) to divide large blocks of text into smaller parts. The `cssStyle` attribute is not obligatory and allows to specify the formatting of elements in (X)HTML. By default the title is displayed in bold type and in larger font, the description and details in smaller font. For a MIDlet, a menu with the entries "Go to title", "Go to description", and "Go to details" is generated (similarly to the "Previous/Next" menu from the `PaginationTag`) and can be used for navigation. The `midpElID` attribute is applied to create forms' names.

- **TableTag, RowTag, CellTag**

syntax:

```
<mitl:table bgcolor="" columns="" border="" cellpadding="" cellspacing=""
midpElID="" cssStyle="">

<mitl:row bgcolor="" width="" height="" cssStyle="">

<mitl:cell bgcolor="" width="" height="" cssStyle="">
```

The `TableTag` draws a table in WML, XHTML, and HTML with a specified background color and border size. The values for cellpadding (specifies the space between content of a cell and its border) and cellspacing (controls the space between table cells) elements can also be given. For WML it is necessary to provide the number of columns.

The `RowTag` generates rows in a table. The background color and the width and height of the row can be provided. The `CellTag` produces cells in a row. Similarly to the `RowTag`, the height and width as well as the background color for each cell may be specified. For all tags, the `cssStyle` attribute can be used for formatting in (X)HTML.

In Java ME, there is no such high-level component like a table. Therefore, `TableItem` and `SimpleTableModel` from the `org.netbeans.microedition.lcdui` package were used to generate a table with rows and cells. The `TableItem` can be used with a `Form` and the content can be provided or modified through the `SimpleTableModel`. The table consists of a header and can potentially be larger than a screen – the user has to use a cursor to scroll in both directions. Since the device properties are retrieved in the `DocTag` and propagated further to all tags, the dynamically created tables do not exceed the screen size. The `midpElID` attribute is used to generate a name for the `TableItem` object. An exemplary table generated with the help of the `TableTag`, `RowTag`, and `CellTag` is displayed in figure 8.13.

¹²⁷ Cf. <http://java.sun.com/j2se/1.5.0/docs/api/java/util/StringTokenizer.html>.

Order ID	Product Name	Delivery date
AB1001	IntelliStation Pro	23.08.2006
AB7002	p5 505 Express	27.11.2006
AB8002	p5 505Q Express	12.11.2006
AB9001	eServer p5 510	23.08.2006
AC2002	System p5 185	21.10.2006
AB4001	IBM ThinkPad	20.08.2006
AB3002	IntelliStation Tower	21.09.2006

Figure 8.13: Table generated by MITL library (Java ME application)

- **HyperlinkTag**

syntax: `<mitl:hyperlink href="" name="" cardID="" midpElID="" accesskey="" cssStyle="">`

The `HyperLinkTag` produces link elements in HTML, WML, and XHTML. The required attribute is the URL of a new document to which the link leads. The `name` attribute specifies an anchor that may be used as a destination of another link. The scope of this attribute is the current document. The `accesskey` attribute allows assigning a special character to this link and is helpful in the navigation process on mobile devices. The `midpEl` attribute provides a name for the Java ME element to which the user should be redirected. The `cardID` attribute is necessary if the link should lead to a WML card that is part of another document. The `cssStyle` attribute describes the formatting of the link in (X)HTML.

In Java ME applications, UI elements representing links are not available. However, the user has the possibility to go from one screen to another with the help of commands. The `HyperLinkTag` generates a menu (similarly to the navigation or pagination menu) that encompasses all actions that should lead to a redirection to a new form or component. The `midpElID` attribute gives the name of the element to which the user can navigate.

- **TickerTag**

- **syntax:** `<mitl:ticker text="" midpElID="" scroll="" align="">`

The `TickerTag` creates a Java ME ticker that displays a text scrolling across the top of the display. The `midpElID` attribute specifies the name of the `Ticker` object, in the `text` attribute the displayed content should be provided. In HTML/XHTML and WML a `marquee` element is generated. The `scroll` and `align` attributes are used only in (X)HTML. They specify the scrolling direction and the alignment of a marquee, respectively.

- **ImageTag**

syntax: `<mitl:image src="" alt="" name="" toFormat="[array]" align="" width="" height="">`

The `ImageTag` is able to transform images and to adapt them to device features basing on the capabilities of the JIMI library. The `src` attribute specifies the name of the image and, eventually, its path. The `name` attribute specifies a new name under which the transformed image should be saved¹²⁸. If a conversion is necessary and no preferences were passed in the `toFormat` attribute, the image is converted by default to the first format supported by a browser (e.g. WBMP, JPG, PNG). The conversion takes into account the information about the display capabilities of a device such as its screen size, the supported color depth, etc. If no graphical format is supported, a textual description from the `alt` attribute is rendered. Additionally, it is possible to specify the alignment of the image, its width and height (the `align`, `width`, and `height` attributes, respectively).

For Java ME applications, an `ImageItem` object is generated. In the constructor of this object, a label (as provided in the `alt` attribute), an `Image` object (the `src` attribute), a layout parameter (the `align` attribute converted to the constants from table 8.16), and an alternative text string are specified, e.g.:

```
Image_1 = new ImageItem
("Some text describing the image: ", Image.createImage ("/someImage.png"),
 ImageItem.LAYOUT_CENTER, "Some description");
```

Constant	Description
LAYOUT_CENTER	The image is centered horizontally.
LAYOUT_DEFAULT	A device-dependent default formatting is applied to the image.
LAYOUT_LEFT	The image is left aligned.
LAYOUT_NEWLINE_AFTER	A new line will be started after the image is drawn.
LAYOUT_NEWLINE_BEFORE	A new line will be started before the image is drawn.
LAYOUT_RIGHT	The image is aligned to the right.

Table 8.16: `ImageItem` layout constants [Top102, p. 418]

The `ImageTag` allows for simple transformations. In order to improve its possibilities, the `ImageTag` library from Jakarta's sandbox can be used [Jaka05]. The tags available in this library and their short descriptions are presented in table 8.17. Unfortunately, the generated markup is limited to the HTML. Therefore, the `Image` tag from the library was rewritten and generates XHTML, WML, and Java ME code. The attributes of this new tag are exactly the same as in the original version. The modified `ImageTag` library was also enclosed in the MITL library.

¹²⁸ If the `name` attribute is missing, the tag saves the image under the same name but with different file extension, e.g. png instead of jpg.

Tag	Description
image	It is the parent tag, all other tags are enclosed inside it. It loads, saves and prints the HTML <code>img</code> tag.
resize	Scales/resizes an image and can be applied to generate thumbnails or to zoom into an image.
overlay	Overlays an image on the parent image. It can paint transparent images. Additionally, a color can be specified to make the pixels transparent.
transparency	Makes the image translucent, supports 255 levels of transparency.
rotate	Rotates the image by a given amount of degrees and changes its dimensions if necessary.
border	Adds a border with a specified thickness and color to an existing image.
grayscale	Converts a color image to grayscale.
crop	Crops or cuts-out a fragment of a bigger image.
text	Writes or paints a text/a string on the image.

Table 8.17: Tags in ImageTag library¹²⁹

Listing 8.16 shows a simple example of using Jakarta's ImageTag library. An image is loaded, scaled by 50 percent and rotated in the anti-clockwise direction by 10 degrees. Subsequently, a border with a thickness of 2 pixels on the sides and 3 pixels on the top and bottom is added. The color of this border is specified in the `color` attribute.

```

1. <img:image
2.     src="/splash.jpg"
3.     name="splash-thumb.jpg"
4.     attributes="alt='A thumbnail'">
5.     <img:resize scale="50%" />
6.     <img:rotate degrees="-10" />
7.     <img:border width="2" height="3" color="FFCC00"/>
8. </img:image>

```

Listing 8.16: Use of Jakarta's ImageTag library

- **FormTag, FieldTag**

syntax:

```

<mitl:form name="" href="" method="" midpElID="" toMIDPEl="">
<mitl:field type="" name="" value="" attr="[array]" midpElID="" imgMIDP=""/>

```

The `FormTag` together with `FieldTags` generate a form with different form elements in (X)HTML, WML and Java ME. The `FormTag` requires three attributes: the `name` of the form, the URL of the page the user data will be passed to (the `href` attribute), and the HTTP method for sending data to the

¹²⁹ Cf. <http://jakarta.apache.org/taglibs/sandbox/doc/image-doc/index.html>.

URL (the `method` attribute, the possible values are `post` or `get`). In (X)HTML the generated output of the `FormTag` `<mitl:form name="myForm" href="form_action.jsp" method="get">` will be as follows: `<form name="form1" action="form_action.jsp" method="get"></form>`. In WML, the same `FormTag` will produce:

```
<do type="accept" label="myForm">
  <go href="form_action.jsp"/>
</do>
```

The `FieldTag` generates different form elements such as input text boxes, text areas, buttons, etc. The `name` attribute specifies the name of the element, the `value` attribute indicates its value and the `attr` attribute can contain an array of additional parameters (possible values are e.g. `multiple` for selecting more elements, `size` for the size of an element, `selected` for the selected element, `disabled` for a disabled one, etc.). The `type` attribute allows specifying the type of an element. Allowed values match with specific (X)HTML/WML elements and are as follows:

- `option` for a drop-down list in (X)HTML, WML
- `optgroup` for an option group in (X)HTML, WML
- `text/password` for a text field/password field in (X)HTML, WML
- `textarea` for a textarea (matches with `textbox` with a maximum length of 30 characters in WML)
- `submit` for a submit button (only for (X)HTML and Java ME, since appropriate buttons for WML are produced by the `FormTag`)
- `reset` for a reset button (only in (X)HTML and Java ME)
- `fieldset` for creating a fieldset in (X)HTML, WML
- `radio` for a radio button (for WML it generates the same output as `option`)
- `check` for a checkbox (for WML it generates the same output as `option`).

For Java ME applications, the `FormTag` will produce a new `Form` object. The `FieldTag` elements can match with two types of UI elements that can be contained in a form: a `TextField` with constant values `TextField.ANY` or `TextField.PASSWORD`¹³⁰ (for the `text/password` type) and a `ChoiceGroup` (`option/optgroup/radio/check` types). The constructor of the `TextField` object takes four values: a label (the `name` attribute), initial text (enclosed in the tag), a maximum text size (the `attr` attribute or default value), and constraints that indicate the type of input allowed (ANY/PASSWORD depending on the element's type). Additionally, the `reset` and `submit` types will produce appropriate commands, placed at the bottom of the form and respective `CommandListener` objects. The user will not be redirected to the URL specified but to the element specified in the `toMIDPEL` attribute. The `midpElID` attribute located in the `FormTag` and `FieldTag` specifies the names of the generated MIDlet elements.

The `ChoiceGroup` [TopI02, p.404] is an MIDP element that enables the choice between different elements (displayed as text and eventually image) in a `Form`. `ChoiceGroups` can be of two different

¹³⁰ The `PASSWORD` constant lets the user enter a password, where the entered text is masked. The `ANY` constant allows any ext to be added.

types: `EXCLUSIVE` or `MULTIPLE`. The `EXCLUSIVE` constant specifies that a `ChoiceGroup` can have only one element selected at the same time, the `MULTIPLE` constant allows the selection of multiple elements. The `option/optgroup/radio/check` types contained in the `type` attribute generate an empty constructor of the `ChoiceGroup` object and add new elements to it using the `append(String stringPart, Image imagePart)` method. The `imgMIDP` attribute is not obligatory and can contain an image for the `ChoiceGroup` component. In order to detect the state changes of the `ChoiceGroup` element, an `ItemStateListener` with the `itemStateChanged(Item item)` method is automatically generated and added to the form.

- **ListTag, ListElement Tag**

syntax: `<mitl:list listArt="ol/ul" listType="A/a/I/i/1" cssStyle="" midpElID="">`
`<mitl:listElem elType="A/a/I/i/1/disc/square/circle" cssStyle="">`

The `ListTag` and `ListElementTag` generate lists and their elements in (X)HTML, WML, and Java ME. In (X)HTML the `listArt` attribute specifies the type of the list – ordered (`ol`) or unordered (`ul`). The `ListElementTag` generates `` tags that define the start of a list item. The `elType` and `listType` attributes provide information about the type of the list and can have one of the following values: `A`, `a`, `I`, `i`, `1` for both tags and `disc`, `square`, or `circle` for the `elType` in the `ListElementTag`. The `cssStyle` attribute allows to specify the formatting of the list and its elements. Since WML does not support lists, the `ListTag` generates a paragraph element and the `ListElementTag` outputs a text ended with a break and inserts appropriate numbering or icon in front of it.

In Java ME, the `ListTag` and `ListElementTag` generate a `List` object. The `List` can contain one or more elements with a text, an optional image, and an optional font for the text. The `List` itself has to possess a title and a policy for the selection of elements. Three options are possible - only one element can be selected (`Choice.EXCLUSIVE`), multiple elements can be selected (`Choice.MULTIPLE`), or the currently highlighted element can be selected (`Choice.IMPLICIT`). For the automatically generated list, the `Choice.IMPLICIT` constant is used. The `List` object is not placed on the `Form` because it inherits from the `Displayable` and not from the `Item` class. The `midpElID` attribute is applied to build a list's name.

- **MobileNavigationMenuTag, MenuItemTag**

syntax: `<mitl:mobNavMenu colNum="" mode="table/list">`
`<mitl:menuItem image="" link="">`

The `MobileNavigationMenuTag` and the `MenuItemTag` generate a menu for mobile devices in (X)HTML and WML. The `colNum` attribute specifies the number of columns for the menu. By default this number is set to 2. The `mode` attribute provides information about the presentation style of the navigation menu. It may be displayed as a list of links or as a table (links are displayed by default). The `MenuItemTag` identifies menu items - for each item an image (the `image` attribute) and a link (the `link` attribute) can be provided. The images together with links are then displayed as items in a list or they are placed in a table.

- **WebLTag**

syntax: <mitl:webL script="" args="[array]">

The `WebLTag` invokes a script written in Compaq's Web Language. The script has to be prepared by a developer and is not generated automatically. In the `script` argument the name of the WebL program is specified (e.g. "MyScript.webl"). The `args` array can contain many parameters that are passed to the WebL script, e.g. the name of the file to which the results will be saved, values for variables, etc. These values are passed to the script using their indexes.

- **MobileTableConversionTag**

syntax: <mitl:tableConv tableType="timetable/lengthway/sideway" attr="X/Y">

The `MobileTableConversionTag` transforms tables to formats that fit on small screens of mobile devices. Three basic types of tables can be distinguished: timetable, lengthways list, and sideways list [cf. MaYaNa01]. The timetable possesses attributes for rows and columns (`AttributeX` and `AttributeY`, respectively). Sometimes meta attributes (`MX`, `MY`) can also be found. They provide information about the attributes. The schema representing a timetable is shown in figure 8.14a. The lengthways list consists of a first row specifying the attributes and further rows providing values for these attributes. The schema of this type is shown in figure 8.15a. The sideways list is a transposed lengthways list. The attributes are placed in the first columns, whereas the values of these attributes appear in further columns. The types of transformed table can be specified in the `tableType` attribute or can be automatically detected.

The `MobileTableConversionTag` reformats the tables according to three schemas. For the timetable, attribute X or attribute Y is selected as the primary attribute according to the value provided in the `attr` parameter. By default attribute Y is chosen. For the first Y attribute all values of X attributes are shown, then the next Y attribute is taken and all values of X attributes are displayed. The result of the transformation is presented in figure 8.14b. If the number of attributes is too large, the table is further split. For example, if the original table consists of 30 X attributes and on one screen only 12 lines can be displayed, the table is transformed to three tables displaying pairs of attributes (cf. table 8.14c). The tables are linked by the previous/next links.

The lengthways list is transformed to a combination of all attributes displayed in the first column and one value for each of them displayed in the second column. Subsequently, in the next tables all attributes and their respective values are presented as shown in figure 8.15b. The same transformation is used for the sideways table. The splitting mechanism is the same as for timetables. The results can be displayed in WML, (X)HTML, and Java ME.

		MX		
		AX1	...	AXn
MY	AY1	value(X1, Y1)	...	value(Xn, Y1)
	AY2	value(X1, Y2)		value(Xn, Y2)

	AYm	value(X1, Ym)	...	value(Xn, Ym)

a) MetaY=MY, AttributeY=AY, MetaX=MX, AttributeX=AX

MetaY:AttributeY1	
MetaX:AttributeX1	value(X1, Y1)
...	...
MetaX:AttributeXn	value(Xn, Y1)



MetaY:AttributeY1	
MetaX:AttributeX1	value(X1, Y1)
...	
MetaX:AttributeX10	value(X10, Y1)
...	

b)

MetaY:AttributeY2	
MetaX:AttributeX1	value(X1, Y2)
...	...
MetaX:AttributeXn	value(Xn, Y2)

MetaY:AttributeY1	
MetaX:AttributeX21	value(X21, Y1)
...	
MetaX:AttributeX30	value(X30, Y1)

c) for n=30

Figure 8.14. Timetable and its transformation [based on MaYaNa01]

AttributeX1	AttributeX2	...	AttributeXn
valueX11)	valueX12	...	valueX1n
...
valueXm1	valueXm2	...	valueXm

a)

AttributeX1	valueX11
AttributeX2	valueX21
...	...
AttributeXn	valueXn1

b)

AttributeX1	valueXm1
AttributeX2	valueXm2
...	...
AttributeXn	valueXmn

Figure 8.15: Lengthways list and its transformation [based on MaYaNa01]

The following two tags were developed exclusively for Java ME applications:

- **J2MEYes/J2MENOtag**
- *syntax*: `<mitl:j2meYes>`, `<mitl:j2meNo>`

Similarly to the `MobileYesTag`, the `J2MEYesTag` indicates that elements enclosed in this tag should be included in the output for devices supporting Java ME. Elements enclosed in the `J2MENOtag` are excluded in the output for devices supporting Java ME.

- **GaugeTag**
- *syntax*: `<mitl:gauge label="" mode="ia/ni" maxValue="" startValue="">`

The `GaugeTag` produces an UI element that can be used to show the progress of an ongoing operation or allow selection of a value from a contiguous range of values. The Java ME component `Gauge` comes in two flavors, interactive and non-interactive (`mode="ia"` and `mode="ni"`, respectively). In the interactive mode the user can adjust the gauge up and down. A non-interactive gauge is updated using a `Timer` event and can be used to represent the progress of a certain task, e.g. retrieving data from a database. A non-interactive gauge can have an "INDEFINITE" maximum value. The `maxValue` attribute indicates the maximum value displayed in a gauge and the `startValue` attribute specifies the beginning value shown in a gauge. If starting and maximum values are not provided, the gauge starts with 1 and ends with 10 for interactive gauge and with INDEFINITE for non-interactive gauge. The `label` attribute provides a caption for a gauge.

8.5 Integrated Development Environment for MITL

One of the main goals of the MITL framework was to provide an Integrated Development Environment that would help a developer to author device-independent applications without being concerned with device-specific issues. The developed IDE (cf. figure 8.16) is a rapid prototyping environment for device-independent authoring based on the Mobile Interfaces Tag Library. It is implemented in Swing and was programmed with help of Oracle JDeveloper 10g [Orac05]. Since Java is a platform-independent language, the IDE can be used on a wide variety of computing platforms (e.g. Unix, Linux, Windows), working in the same way no matter where it runs. The IDE consists of three views:

- a markup source code view
- a tree view depicting the structure of folders on a given computer
- a structure/error view illustrating the hierarchical structure of used tags and indicating the errors that a developer made

The IDE supports rapid development of device-independent applications even for users without previous experience with Mobile Interfaces Tag Library. The author does not have to know the MITL structure in detail. If he/she chooses a tag from the menu "Tags" or an appropriate shortcut from the taskbar, the tag is automatically created and inserted into the document. For all tags, the attributes are displayed explicitly in the form of panels with text boxes. This significantly reduces the time needed to prepare a MITL document, because tag elements do not have to be learned by heart. If the values of

attributes are predefined (e.g. `ul/ol` values for the `ListTag`), they appear in a drop-down list. The user may select one or many of them (for attributes where an array of values is allowed). Additionally, obligatory attributes are marked using red color and an asterisk - the application will prompt user to complete the tag if these are left empty. It is not possible to create and insert a tag into a document without providing obligatory attributes. Moreover, the syntax of a MITL document can be validated, decreasing the possibility of errors.

The IDE offers functionalities that are common to all editors such as opening files, saving changes to a document or saving the document to a file, closing the document and exiting the application. It is possible to create new documents and edit them (undo/redo, cut, copy, and paste functionalities). The user can change the look and feel of the application, i.e. the colors of the IDE elements or the fonts used in MITL documents. The edited MITL files can be arranged vertically, horizontally, or cascaded.

The IDE is integrated with some mobile simulators (Nokia, OpenWave, Yospace) and desktop browsers (Internet Explorer, Opera, and Netscape). The mobile emulators offer a real-time preview of the developed GUI on different mobile handsets. Additionally, the user can check whether his/her Java ME applications run correctly on Java ME-capable devices using the Java ME Wireless Toolkit. It is furthermore possible to start/stop the Apache Tomcat server or any other application server from the IDE. The environment is highly configurable – the user can choose the servers on which the libraries should run as well as the simulators.

The icons representing available tags (as displayed in figure 8.17) are grouped with regard to their functionality on one taskbar (e.g. the `TableTag`, `RowTag`, `CellTag` are placed together and distinctly separated from other tags so that it is easy for the user to locate and apply them), all other icons (e.g. responsible for starting the Web server, viewing files in various browsers, etc.) can be found on another taskbar. All pre-programmed tags and features of the developed IDE can also be accessed by using the drop-down menus. Extensive help with the introduction on how to use the environment, examples of simple and sophisticated documents written in MITL, and answers to typical problems are provided, so that even a new user can quickly write own MITL page.

The implemented validation is based on a Document Type Definition (DTD) document describing the structure of the MITL syntax. It is possible because the MITL language is a well-formed XML. DTD is part of the XML 1.0 recommendation and specifies the tags that can be included in the XML document and their valid arrangements. It defines each allowed element in an XML document, the permitted attributes and their possible values as well as the number of occurrences of each element (zero, one or any number of times). If the user chooses the validation function from the menu bar, a MITL document is validated against the defined DTD using the Xerces parser [ASF05d]. The errors produced by this parser are caught and transformed into human readable error messages displayed in a separate window in the Integrated Development Environment. When the user clicks on the error message, the cursor moves to the code line with the error. The parser is also used to generate the tree structure of a document.

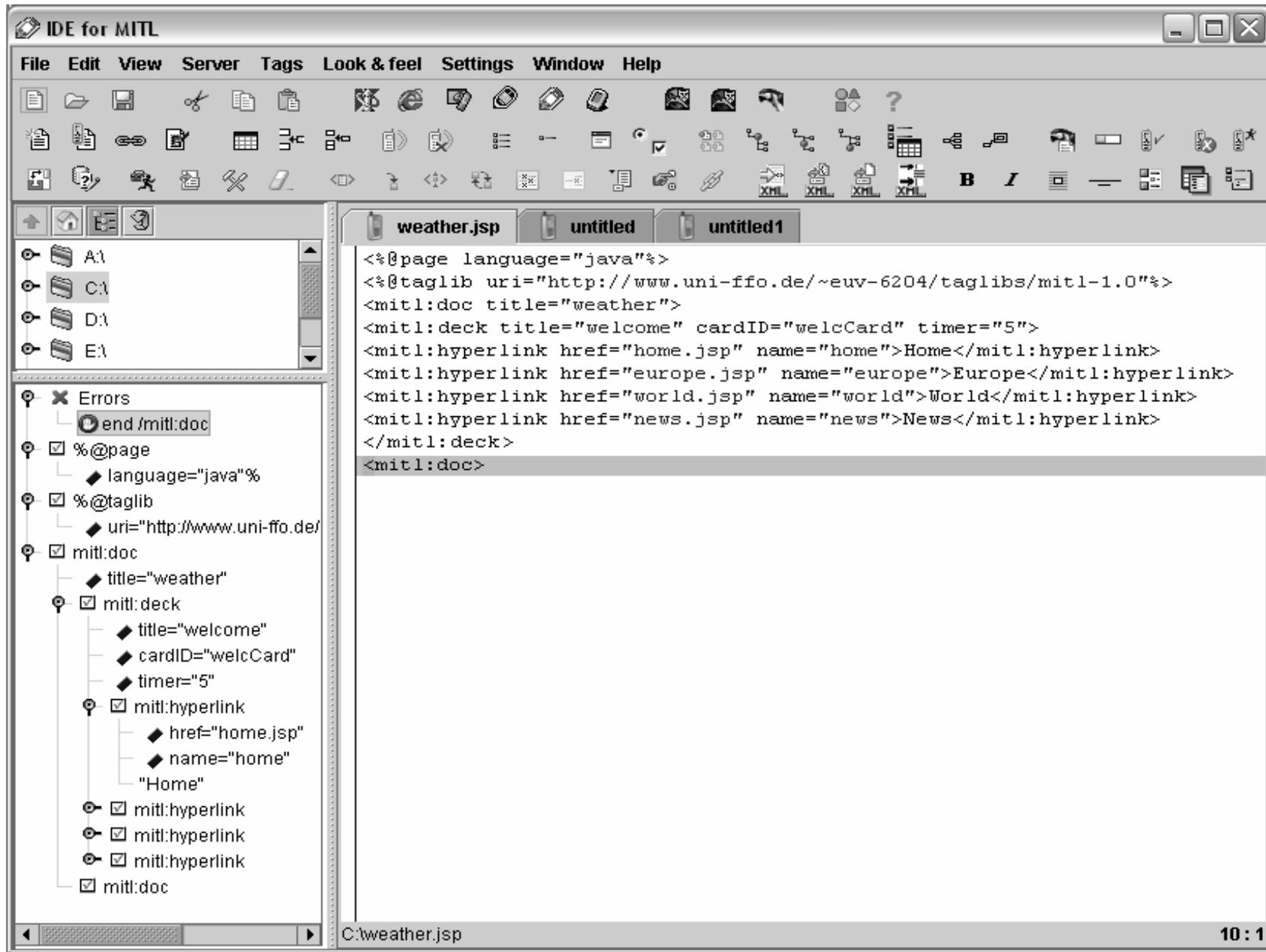


Figure 8.16: Integrated Development Environment for MITL

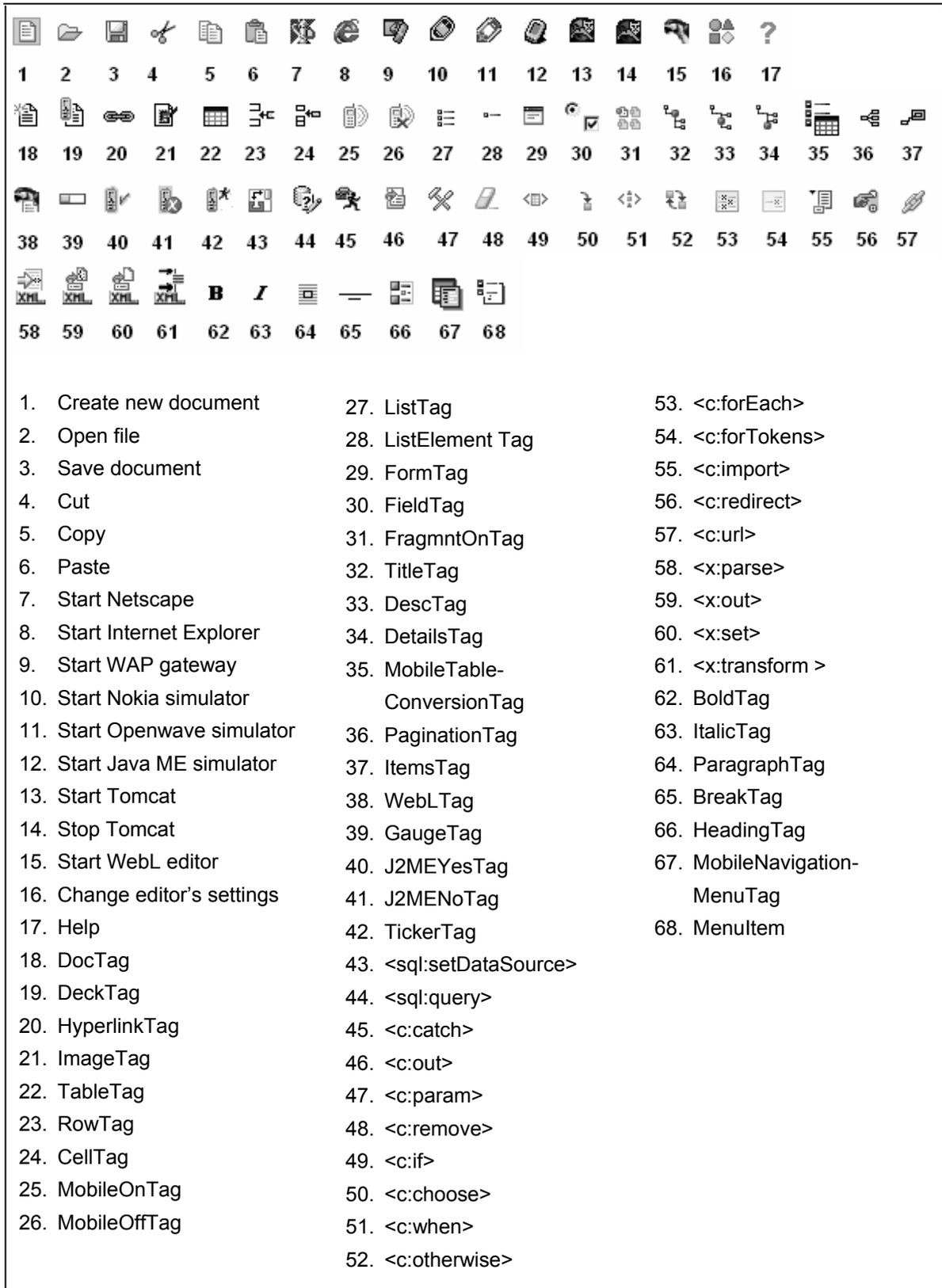


Figure 8.17: Shortcuts from the taskbar of the IDE for MITL

8.6 Usage examples

The MITL library was used to generate simple and complex device-independent applications. This section demonstrates some exemplary applications: a mobile Google service and a weather forecasting service. The mobile Google service uses the content generated by the Google search engine or retrieved with the help of Google tag library. The weather service extracts data about weather prospects from a database or from the Yahoo! Weather RSS feed.

Mobile Google service – parsing a Web page and using Google Tag Library

The powerful search engine implemented by Google (<http://www.google.com> for its English version) allows for searching on an immense amount of information. There is still some confusion about the number of indexed pages. In February 2005 the firm claimed to have more than 8 billion pages indexed, but due to the “index war” between Google and Yahoo, Google stopped publishing the number of pages¹³¹. Instead of this, the company started to claim that its engine is the most comprehensive one and admitted that the counting methodology differs significantly between search engine providers.

Query parameter	Description	Example
site	Restricts results to sites within the specified domain	“site:uni-ffo.de mobile” will find all sites containing the word <i>mobile</i> , located within the *.uni-ffo.de domain
intitle	Restricts results to documents whose titles contain the specified phrase	“intitle:handsets mobile” will find all sites with the word <i>handsets</i> in the title and <i>mobile</i> in the text
allintitle	Restricts results to documents whose titles contain all the specified phrases	“allintitle:mobile devices” will find all sites with the words <i>mobile</i> and <i>devices</i> in the title
inurl	Restricts results to sites whose URLs contain the specified phrase	“inurl:mobile device” will find all sites containing the word <i>device</i> in the text and <i>mobile</i> in the URL
allinurl	Restricts results to sites whose URL contains all the specified phrases	“allinurl:mobile devices” will find all sites with the words <i>mobile</i> and <i>devices</i> in the URL
filetype, ext	Restricts results to documents of the specified type	“filetype:pdf mobile computing” will return PDFs containing the words <i>mobile computing</i>
numrange	Restricts results to documents containing a number from the specified range	“numrange:1-100 mobile” will return sites containing a number from 1 to 100 and the word <i>mobile</i>
link	Restricts results to sites containing links to the specified location	“link:www.w3c.org” will return documents containing one or more links to <i>www.w3c.org</i>
inanchor	Restricts results to sites containing links with the specified phrase in their descriptions	“inanchor:wireless” will return documents with links whose description contains the word <i>wireless</i>
allintext	Restricts results to documents containing the specified phrase in the text, but not in the title, link descriptions or URLs	“allintext:“mobile commerce”” will return documents which contain the phrase <i>mobile commerce</i> in their text only

Table 8.18: Search query parameters in Google [Piot05, p.3]

¹³¹ More information about the “index war” can be found under: <http://battellemedia.com/archives/001889.php>.

Google offers a number of options that can be considered in a search query and that can facilitate the search. The most important query operators are summarized in table 8.18. In addition to specialized search tools (e.g. image search, video search, book search, etc.), the firm offers a set of products for desktop computers such as Picasa for finding and editing pictures or Google talk for instant messaging and voice calls¹³².

The developed device-independent application performs a Google search for a provided search term and displays the results returned by the engine on different devices. This example does not aim at replacing Google mobile search service; it should merely illustrate how to use the Mobile Interfaces Tag Library for the presentation of content retrieved from existing Web pages. The results obtained from the search engine are displayed in a browser. They are extracted and saved in XML with the help of Web Language script displayed in listing 8.17.

```

1. import Files;
2. import Str;
3. var myQ = ARGS[1];
4. var myNo = ARGS[2];
5. var P = GetURL("http://www.google.com/search",
6. [. q= myQ, num=myNo, hl="en", ie="ISO-8859-1", meta="lr%3Dlang_en" .],
7. [. "Accept-Charset" = "iso-8859-1,*utf-8",Connection = "Keep-Alive",
8. "User-Agent" = "Mozilla/4.04[en] (WinNT; I)",
9. Accept = "image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png,
10. */*","Accept-Language" = "en" .]);
11. var XMLStr = "<?xml version='1.0'? encoding='ISO-8859-2'>";
12. var myWrapper = Elem(P, "a") directlyafter Elem(P,"p");
13. var myWrapper1 = Seq(P, "a br font");
14. var myPosition = 0;
15. var i=0;

16. XMLStr = "<GoogleResults><![CDATA[";
17. every W in myWrapper do
18.   XMLStr = XMLStr + "<result>\n";
19.   XMLStr = XMLStr + "<title>" + Text(W) + "</title>\n";
20.   XMLStr = XMLStr + "<link>" + W.href + "</link>\n";
21.   if i<ToInt(myNo) then
22.     myPosition=Str_IndexOf(".www",Text(myWrapper1[i]));
23.     if myPosition!=-1 then
24.       XMLStr = XMLStr + "<desc>" +
25.         Select(Text(myWrapper1[i]), 0, myPosition+1) + "</desc>\n";
26.     end;
27.   end;
28.   XMLStr = XMLStr + "</result>\n";
29.   i=i+1;
30. end;
31. XMLStr = XMLStr + "]]></GoogleResults>";
32. Files_SaveToFile("C:/Google.xml", ToString(XMLStr));

```

Listing 8.17: Exemplary WebL script for extracting data from Google search results

¹³² Cf. <http://www.google.com/intl/en/options/> for more information about Google products.

In order to retrieve certain elements from a page, it is crucial to know its structure. In this example, the text to be retrieved is contained in a link that is a nested element of a paragraph. Additionally, the retrieved content is also a part of a sequence of `<a>`, `
`, and `` tags. The results brought by the Google search engine are placed in appropriate XML tags (`<title>`, `<link>`, and `<desc>`). Subsequently they are converted to a String, and saved in the file "Google.xml". A fragment of the produced XML file is presented in listing 8.18.

```

1. <GoogleResults>
2. <![CDATA[
3. <result>
4.   <title> Mobile Commerce (m-Commerce) </title>
5.   <link> www.cellular.co.za/mcommerce.htm </link>
6.   <desc> One of the Worlds most popular mobile portals. Over 1 million global
       visitors per month </desc>
7. </result>
8. <result>
9.   <title>Wireless News for IT Managers</title>
10.  <link>http://www.internetnews.com/wireless/</link>
11.  <desc> Daily news about mobile computing including PDAs, cell phones,
       handhelds, enterprise platforms and wireless applications. </desc>
12. </result>
       [...]
13. ]]>
14. </GoogleResults>

```

Listing 8.18: XML generated by the WebL script *Google.webl*

```

1. <%@ page language="java"%>
2. <%@ page errorPage="error.jsp" %>
3. <%@ taglib uri="http://www.uni-ffo.de/~evu-6204/taglibs/mitl-1.0" prefix="mitl"%>
4. <mitl:doc title="m-commerce - Google search" stylesheet="googleCSS.css"
       altStylesheet="google1.css">
5.   <mitl:deck title="m-commerce - Google search" cardID="card1">
6.     <% String [] toFrmt = {"png","jpg","gif"};%>
7.     <mitl:image src="Google.jpg" alt="Google image" name="ImgGoogle"
       toFormat="<%= toFrmt %>"/>
8.     <mitl:form name="search" href="rsltGoogle.jsp" method="post"
       midpElID="frmGoogle" toMIDPEl="frmRsltGoogle">
9.       <mitl:field type="text" name="term" midpElID="textField1"/>
10.      <mitl:field type="submit" value="Google search" midpElID="frmBtn1"/>
11.    </mitl:form>
12.  </mitl:deck>
13. </mitl:doc>

```

Listing 8.19: Mobile Google search service (*google.jsp*)

The first page for the device-independent Google search (*google.jsp*) is presented in listing 8.19. It consists of the Google image and a form containing a text box, where the user can enter the search term and a submit button. The preferred image formats are also provided explicitly in the code. They

have to be passed to the `ImageTag` as an array of `String` objects. This requires adding of a small JSP scriptlet `<% Object [] toFrmt = {"png","jpg","gif"}; %>` that defines an array `toFrmt` and initializes it with some values. This array is subsequently passed to the `toFormat` attribute of the `ImageTag` (expression `<%= toFrmt %>`). The array and the scriptlet are automatically generated by the IDE, the user has to provide the array values.

```

1. <%@ page language="java"%>
2. <%@ page errorPage="error.jsp" %>
3. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/mitl-1.0" prefix="mitl"%>
4. <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
5. <%@ taglib uri="http://java.sun.com/jstl/xml" prefix="x" %>

6. <mitl:doc title="Google" stylesheet="google1.css" >
7.   <mitl:deck title="m-commerce - Google search" cardID="card2">
8.     <mitl:mobileNo>
9.       <% String [] toFrmt = {"png","jpg","gif"};%>
10.      <mitl:img src="Google.jpg" alt="Google image" name="imgGoogle"
           toFormat="<%= toFrmt %>" align="center"/>
11.    </mitl:mobileNo>
12.    <mitl:form name="frmGoogle" href="rsltGoogle.jsp" method="post"
           midpElID="frmGoogle" toMIDPEl="rsltGoogle">
13.      <mitl:field type="text" name="term" value="" midpElID="textField1"/>
14.      <mitl:field type="submit" value="Google search" midpElID="frmBtn1"/>
15.    </mitl:form>
16.    <c:set var="term" scope="session" value="${param.term}"/>
17.    <%Object [] myArray= ${sessionScope.term},"100"}; %>
18.    <mitl:webL script="Google.web1" args="<%= myArr %>"/>
19.    <mitl:pagination>
20.      <mitl:items>
21.        <x:parse var="googleResult">
22.          <c:import url="Google.xml"/>
23.        </x:parse>
24.        <mitl:FragmntOn>
25.          <x:forEach select="$googleResult/result">
26.            <x:set var="title" select="title" />
27.            <x:set var="desc" select="desc" />
28.            <x:set var="details" select="link" />
29.            <mitl:title midpElID="titleEl">${title}</mitl:title>
30.            <mitl:desc midpElID="descEl">${desc}</mitl:desc>
31.            <mitl:details midpElID="detailsEl">${details}</mitl:details>
32.          </x:forEach>
33.        </mitl:FragmntOn>
34.      </mitl:items>
35.    </mitl:pagination>
36.  </mitl:deck>
37. </mitl:doc>

```

Listing 8.20: Mobile Google search service (*rsltGoogle.jsp*)

After entering the search term and pressing the “Google search” button, the user is redirected from the first page (*google.jsp*) to the second page (*rsltGoogle.jsp*) consisting of the Google image, a new search form, and a list of results. The code of the second page is presented in listing 8.20. In order to generate a page with results, the JSTL tag library was used together with the MITL tag library. The `WebLTag` invokes a Web Language script that generated the XML file “Google.xml”, presented in

listing 8.18. The content of this file is stored in the variables `title`, `desc`, and `details`. These variables are then used to associate appropriate fragments of the XML file with corresponding fragmentation tags. The whole content is additionally paginated with the help of the `PaginationTag`. Depending on the device, the results of a search query are displayed in different ways as shown in figures 8.18, 8.19, and 8.20.

For desktop browsers the number of displayed search results amounts to 6, for the presented mobile devices (Nokia 6600 and Java ME simulator) to 3. In Internet Explorer, all parts of the search result are shown together – the title, the description, and the link. In the mobile browser of Nokia 6600 only titles are displayed. They serve as links to respective links' descriptions. In Java ME applications, when the user selects a result (title or description) and chooses the menu entry “Go to details“, he/she can view on the next form a link to a Web page. The fragmentation mechanism takes into account mobile context. Mobile users usually do not want to pay for a long lasting search and browsing through all the found results. They focus rather on reading a short description of a page and, if it matches their requirements, they can note the address of the page and view it at home.



Figure 8.18: Mobile Google search service in XHTML (Nokia 6600)

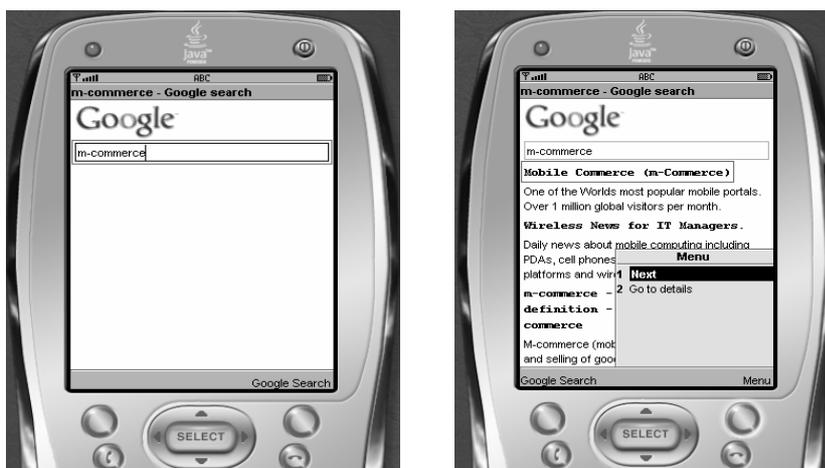


Figure 8.19: Mobile Google search service as Java ME application (Java ME simulator)

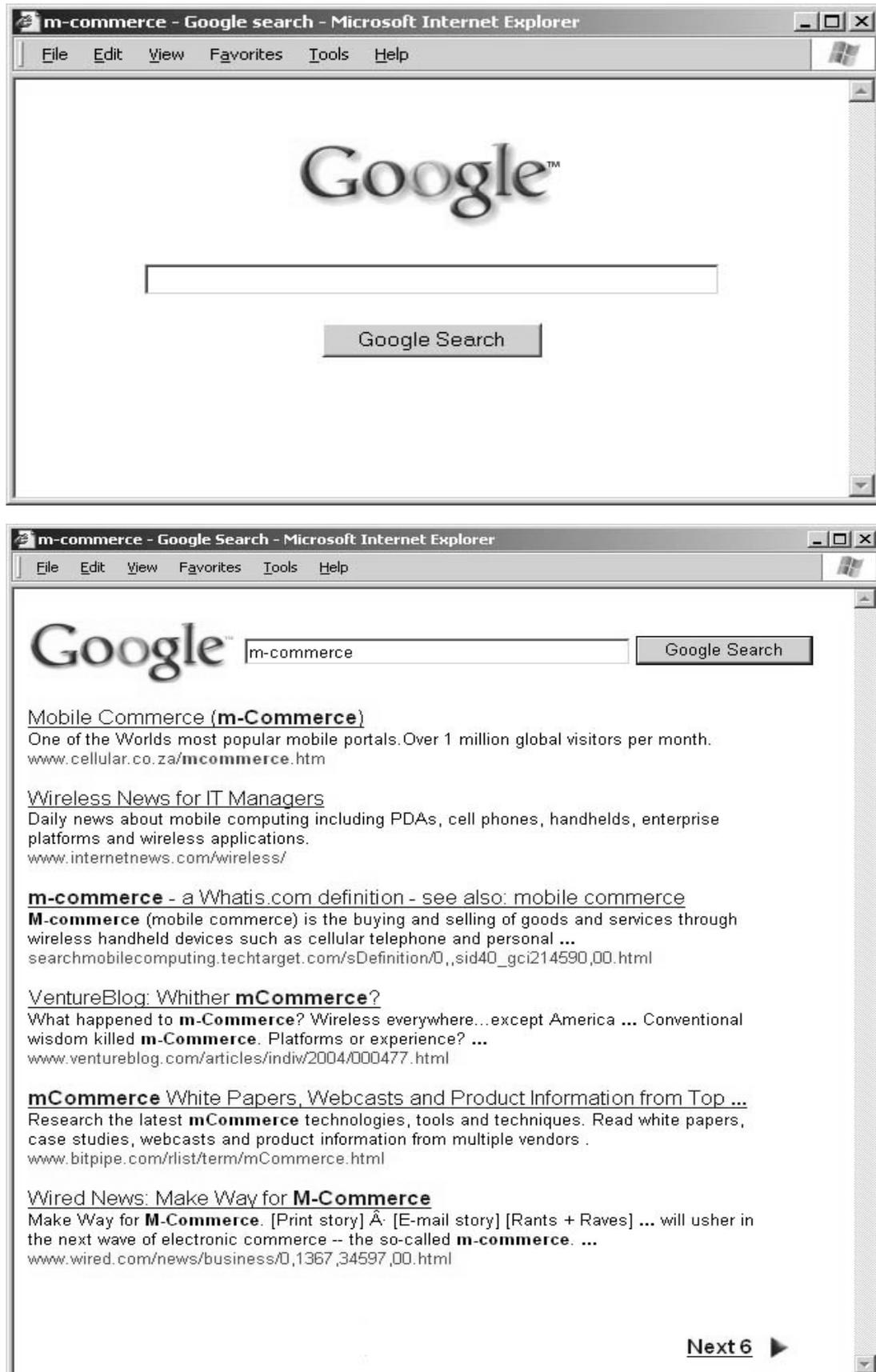


Figure 8.20: Mobile Google search service in HTML (Internet Explorer)

Instead of using Web Language to parse a Web page with some search results returned by Google, it is also possible to use a Google Tag Library [Thau+05]. This tag library is able to perform a standard Google search based on the Google API [Goog06]. It can retrieve a summary, an URL, or a title of a page that matches the search term, enables pagination of results, can show a cached page or give the estimated search time. Table 8.19 summarizes the most important tags available in the Google Tag Library. Listing 8.21 shows an example of device-independent Google search based on this tag library. The elements of the result sets are retrieved with tags from the Google Tag Library (cf. lines 12-16). The returned output matches the search query for the term “e-commerce”. The results are similar to the pages built with the help of Web Language. Exemplary pages are displayed in figure 8.21.

Tag	Description
<code><google:search/></code>	Performs the search.
<code><google:searchResult></code>	Returns the result set.
<code><google:element name="url"/><</code>	Displays the properties of the current result set item. The following <code>name</code> attributes can be used to display different elements: <ul style="list-style-type: none"> - <code>summary</code> - shows a summary of the page (as a text) - <code>URL</code> - displays the URL of the search result (an absolute URL path) - <code>snippet</code> - shows a snippet of the result - <code>title</code> - displays the title of the search result - <code>cachedSize</code> - displays the size of the cached version of the URL
<code><google:next></code> <code><google:previous></code>	Enables pagination of a result set.
<code><google:startIndex/>/</code> <code><google:endIndex/></code>	Displays the index of the first search/last search result in the current result set.
<code><google:estimatedTotal/></code>	Displays the estimated total number of results for the current query.
<code><google:searchQuery/></code>	Displays the text of the current query.
<code><google:searchTime/></code>	Shows the total server time used to return the search results (in seconds).
<code><google:cachedPage/></code>	Shows Google's cached Web pages.
<code><google:spelling/></code>	Displays Google's spelling suggestions.

Table 8.19: Tags from Google Tag Library [Thau+05]



Figure 8.21: Results of the search with Google Tag Library on Motorola V980 and Sharp GX-10 phones

```

1. <%@ page language="java"%>
2. <%@ page errorPage="error.jsp" %>
3. <%@ taglib uri="http://google-taglib.sourceforge.net/google-taglib" prefix="google" %>
4. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/mitl-1.0" prefix="mitl"%>
5. <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>

6. <% String title=Search for <google:searchQuery/> %>
7. <mitl:doc title=<%=title%> stylesheet="myGoogle.css">
8.   <mitl:deck title=<%=title%> cardID="card1">
9.     [...]
10.    <mitl:pagination>
11.      <mitl:items>
12.        <google:estimatedTotal/> results for e-commerce <mitl:break/>
13.          <google:searchResult>
14.            <google:element name="title-url"/>
15.          </google:searchResult>
16.        </mitl:items>
17.      </mitl:pagination>
18.    <google:search/>
19.  </mitl:deck>
20. </mitl:doc>

```

Listing 8.21: Code for Google search with MITL and GoogleTag Library

Weather forecasting services – interaction with a database, application of RSS feeds

The next examples demonstrate the possibility of database interaction in MITL documents and use of the information delivered by RSS feeds. Both applications are weather forecasting services created with the MITL library and rendered differently in various browsers.

In the first example, data was retrieved from a database with the help of DBTags library introduced in section 8.3.4. Listing 8.22 shows fragments of code responsible for the establishment of a connection with a database (cf. line 3), sending a query to this database (cf. lines 4-6) and retrieving records from it (cf. lines 7-11). The following data was obtained from the table named “forecast” and displayed as follows: name of a day, minimum and maximum temperature and the amount of rain (cf. figure 8.22 and 8.23, two last screens). The obtained information is presented in the form of HTML pages and as XHTML, WML, or Java ME application for mobile devices. The content of one big HTML page, shown in figure 8.24, was split into many pages for devices supporting XHTML, WML, and Java ME with regard to the screen size. This operation was performed with the `DeckTags`. They are ignored in the case of desktop browsers and allow to split one page into a chosen number of smaller pages (cards in WML or set of linked pages in XHTML). All pictures are available only in the HTML and Java ME versions of the application. They were excluded from the mobile pages with the help of the `MobileNoTag`.

```

1. <%@ taglib prefix="c" uri="/WEB-INF/tld/c.tld" %>
2. <%@ taglib uri="/WEB-INF/tld/sql.tld" prefix="sql" %>
   [...]
3. <sql:setDataSource var="connDB" driver="sun.jdbc.odbc.JdbcOdbcDriver"
   url="jdbc:odbc:weather"/>
   [...]
4. <sql:query var = "forecast" dataSource="{connDB}">
5.     SELECT * from forecast
6. </sql:query>
   [...]
7. <c:forEach var="row" items="{forecast.rows}">
8.     <c:out value="{row.day}"/><br/>
9.     <c:out value="{row.temp_min}"/> / <c:out value="{row.temp_max}"/> C <br/>
10.    <c:out value="{row.rain}"/> mm<br/>
11. </c:forEach>
   [...]

```

Listing 8.22: Fragments of code for the interaction with database

The pagination was performed with the `PaginationTag`, whereas the number of displayed elements depends on the screen size of a device and calculated number of visible lines. Figures 8.22 and 8.23 present the weather forecasting service on Siemens SX1 (XHTML) and Motorola C975 (WML). In both cases the result set was divided into parts consisting of 5 elements.



Figure 8.22: XHTML presented in Siemens SX1 browser

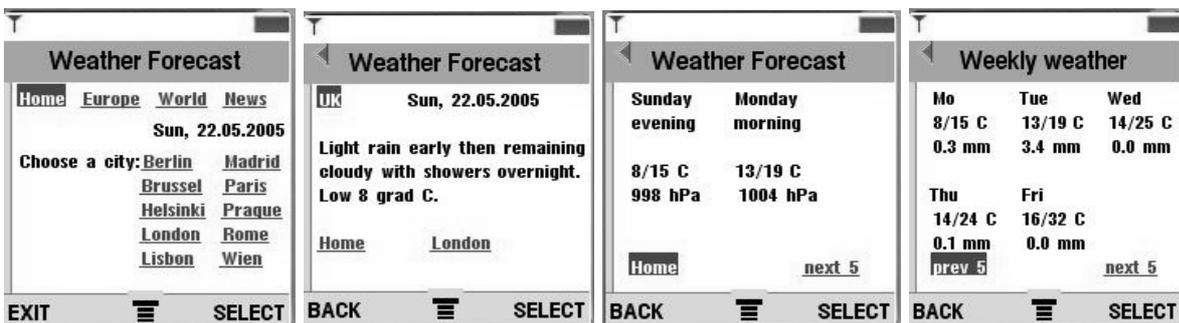


Figure 8.23: WML displayed in Motorola C975 browser

Weather Forecast - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Home Europe World News

< Sunday >



United Kingdom Sunday, 22.05.2005

Weather for today :
Light rain early then remaining cloudy with showers overnight.
Low 8 °C. Winds SW at 5 to 8 km/h. Chance of rain 70%.

Sunset 20:54 (BST) Pressure 1017 hPa
Sunrise 5:00 (BST)

London

Sunday evening Monday morning

8/15°C 998 hPa	13/19°C 1004 hPa
-------------------	---------------------

Satellite



NET7 22 MAY 2005 1500 B2

copyright © 2005 EUMETSAT

Choose a city :

- [Berlin](#)
- [Brussel](#)
- [Helsinki](#)
- [London](#)
- [Lisbon](#)
- [Madrid](#)
- [Paris](#)
- [Prague](#)
- [Rome](#)
- [Wien](#)

Weather Forecast London - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Home Europe World News

LONDON Sunday, 22.05.2005

Detailed forecast

Now : Monday morning:

8 / 15°C 998 hPa	Rain : 0.3 mm Wind : 13 km/h	13 / 19°C 1004 hPa	Rain : 0.9 mm Wind : 21 km/h
---------------------	---------------------------------	-----------------------	---------------------------------

Overview for the coming week:

Monday, 23.05.2005	Tuesday, 24.05.2005	Wednesday, 25.05.2005	Thursday, 26.05.2005
8 / 15°C 1006 hPa	13 / 19°C 990 hPa	14 / 25°C 1007 hPa	14 / 24°C 1008 hPa
Rain : 0.3 mm Wind : 13 km/h	Rain : 3.4 mm Wind : 27 km/h	Rain : 0.0 mm Wind : 22 km/h	Rain : 0.1 mm Wind : 19 km/h
Friday, 27.05.2005	Saturday, 28.05.2005	Sunday, 29.05.2005	Monday, 30.05.2005
16 / 32°C 1003 hPa	19/27°C 1007 hPa	12/22°C 1009 hPa	12/19°C 1013 hPa
Rain : 0.0 mm Wind : 21 km/h	Rain : 0.0 mm Wind : 21 km/h	Rain : 2.0 mm Wind : 16 km/h	Rain : 0.6 mm Wind : 31 km/h

Figure 8.24: HTML version of weather forecasting service (in Internet Explorer)

Mobile services do not have to be built from scratch. A weather forecasting service can be based, for example, on the Yahoo! Weather RSS feed (<http://developer.yahoo.com/weather/>). Listing 8.23 presents an exemplary RSS 2.0 document with information about the weather in Frankfurt/Oder on the 29th of August 2006 and the following two days. In addition to traditional tags that can be encountered in each RSS 2.0 feed (`<title>`, `<link>`, `<description>`, `<language>`, `<ttl>`, etc.), the Yahoo! Weather feed includes two namespaces with own tags (the `yweather` and `geo` namespaces). These proprietary tags contain the most important weather information, e.g. data about wind, temperatures, sunrise and sunset, etc. (cf. lines 11-14 and 39-40 of listing 8.23). Figure 8.27 shows the original version of Yahoo! Weather service in HTML. Only some parts of the information provided in HTML are made available in the RSS feed.

In order to use the feed for a weather service, the RSS file has to be parsed and the relevant data should be extracted. This can be accomplished with any of the RSS libraries introduced in section 8.3.6 or with the help of the XML processing library. The RSS Utilities tag library makes it possible to retrieve data from standard RSS tags. It does not, however, support proprietary tags and namespaces. Listing 8.24 shows an example of extracting information from the Yahoo! Weather RSS feed. The RSS feed can be obtained from the address: <http://xml.weather.yahoo.com/forecast/rss?p=GMXX0185&u=c>, where the `p` parameter defines the location (city) and the `u` parameter the measurement for temperature (`c` stands for Celsius, `f` for Fahrenheit). Weather forecast is retrieved from the `<description>` tag nested in the `<item>` tag (lines 29-39 in listing 8.23). For this purpose, the `<rss:itemDescription feedId="yahooFeed"/>` tag is used. The `feedID` attribute refers to the URL of the Yahoo! Weather feed specified in the `<rss:feed>` tag. Figures 8.25 and 8.26 present the Yahoo! Web Service developed with the help of RSS Utilities tag library and the MITL tag library. The user can enter a location, for which he needs a forecast, in a textbox. If Yahoo! Weather provides an RSS feed for this location, the feed is retrieved, the forecast is extracted (cf. listing 8.24) and slightly reformatted. Additionally, navigation elements are generated.

The XML processing tag library can also be helpful in the extraction of data. It uses XPath expressions to select relevant fragments from the XML file. Listing 8.25 presents an exemplary code for the retrieval of information contained in tags from the `yweather` namespace. Information about wind (lines 8-11), sunrise and sunset (lines 12-15), and the forecasts for next two days (lines 16-20) is extracted. The data is placed as attributes' values and has to be read from tags preceded by a namespace. In order to select a tag with a namespace, the `root//*[name()='namespace:tag']` expression has to be used. The `root//*[name()='namespace:tag']` allows to select any tags that are below the root element, independently of the number of nested tags. The expression `"name()='namespace:tag'"` matches all tags with a specified namespace and tag name. The `@` symbol represents a tag's attribute. To extract the value of an attribute the `@attribute_name` (e.g. `@sunrise`) expression has to be applied. Therefore lines 12-15 in listing 8.25 allow to select all values of the attributes `sunrise` and `sunset` that are contained in the `yweather:astronomy` tag. This example demonstrates that working with XML fragments requires some knowledge of the XPath specification and may be difficult for inexperienced content authors.

```

1. <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2. <rss version="2.0" xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0"
   xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
3. <channel>
4.   <title>Yahoo! Weather - Frankfurt, GM</title>
5.   <link>http://us.rd.yahoo.com/dailynews/rss/weather/Frankfurt__GM/
   *http://xml.weather.yahoo.com/forecast/GMXX0185_c.html</link>
6.   <description>Yahoo! Weather for Frankfurt, GM</description>
7.   <language>en-us</language>
8.   <lastBuildDate>Tue, 29 Aug 2006 10:00 pm CEST</lastBuildDate>
9.   <ttl>60</ttl>
10.  <yweather:location city="Frankfurt" region="" country="GM" />
11.  <yweather:units temperature="C" distance="km" pressure="mb" speed="kph" />
12.  <yweather:wind chill="12" direction="260" speed="18" />
13.  <yweather:atmosphere humidity="92" visibility="32000" pressure="1003" rising="1" />
14.  <yweather:astronomy sunrise="6:07 am" sunset="7:57 pm" />
15.  <image>
16.    <title>Yahoo! Weather</title>
17.    <width>142</width>
18.    <height>18</height>
19.    <link>http://weather.yahoo.com/</link>
20.    <url>http://us.il.yimg.com/us.yimg.com/i/us/nws/th/main_142b.gif</url>
21.  </image>
22.  <item>
23.    <title>Conditions for Frankfurt, GM at 4:00 pm CEST</title>
24.    <geo:lat>52.34</geo:lat>
25.    <geo:long>14.55</geo:long>
26.    <link>http://us.rd.yahoo.com/dailynews/rss/weather/Frankfurt__GM/
   *http://xml.weather.yahoo.com/forecast/GMXX0185_c.html</link>
27.    <pubDate>Tue, 29 Aug 2006 10:00 pm CEST</pubDate>
28.    <yweather:condition text="Mostly Cloudy" code="27" temp="12"
   date="Tue, 29 Aug 2006 10:00 pm CEST" />
29.    <description><![CDATA[
30.      <br />
31.      <b>Current Conditions:</b><br />
32.      Mostly Cloudy, 12 C<BR /><BR />
33.      <b>Forecast:</b><BR />
34.      Tue - Light Rain. High: 14 Low: 10<br />
35.      Wed - Light Rain. High: 17 Low: 11<br />
36.      <br />
37.      <a href="http://us.rd.yahoo.com/dailynews/rss/weather/Frankfurt__GM/*
   http://xml.weather.yahoo.com/forecast/GMXX0185_c.html">Full Forecast at
   Yahoo! Weather</a><BR/>(provided by The Weather Channel)<br/>
38.    ]]></description>
39.    <yweather:forecast day="Tue" date="29 Aug 2006" low="10" high="14"
   text="Light Rain" code="11" />
40.    <yweather:forecast day="Wed" date="30 Aug 2006" low="11" high="17"
   text="Light Rain" code="11" />
41.    <guid isPermaLink="false">GMXX0185_2006_08_29_22_0_CEST</guid>
42.  </item>
43. </channel>
44. </rss>

```

Listing 8.23: RSS feed for Yahoo! Weather



Figure 8.26: Yahoo! Weather forecasting service on Sony Ericsson T610 (WML version)

http://weather.yahoo.com - Frankfurt Weather Forecasts on Yahoo! Weather - Microsoft Internet Explorer

Yahoo! My Yahoo! Mail Make Y! your home page Search: Web Search

YAHOO! NEWS Weather Sign In New User? Sign Up Weather Home - Help

Home U.S. Business World Entertainment Sports Tech Politics Science Health Travel Most Popular

Photos Opinion Local News Odd News Comics Weather Full Coverage Video/Audio Kevin Sites Site Index

Search: All News & Blogs Search Advanced

Frankfurt Weather Add Frankfurt Weather to Your My Yahoo! Page **RSS**

The Weather Channel weather.com Change Location (About My Yahoo! and RSS - RSS Help)

Weather > Europe > Germany > Frankfurt F° | C°

Current conditions as of 12:00 am CEST

Mostly Cloudy

Feels Like: 11°
 Barometer: 1004 mb and rising
 Humidity: 92%
 Visibility: Unlimited
 Dewpoint: 10°
 Wind: W 18 kph
 Sunrise: 6:07 am
 Sunset: 7:57 pm

11° High: 14° Low: 10°

» Detailed Forecast
 » Records & Averages
 » Get Yahoo! Weather on your desktop

MORE FROM WEATHER.COM

- Tracking Ernesto's Path
- Local Mosquito Levels
- Flight Delays
- Grilling Out Forecast
- Schoolday Weather
- Download Desktop Weather

ADVERTISEMENT

1/2 Price Days & Sale

Sale Ends Soon!

LAMPS PLUS SHOP NOW

TODAY	TOMORROW	THU	FRI	SAT	6-10 DAY
					Extended Forecast
Light Rain	Light Rain	Showers	Showers	Showers	
High: 14° Low: 10°	High: 17° Low: 11°	High: 19° Low: 11°	High: 20° Low: 11°	High: 23° Low: 11°	

Get Alerts: Mobile Snowfall Alerts Weather Bulletins

Featured Forecasts at weather.com:
 Hurricane Central | Mosquito Activity | Labor Day Forecast

Figure 8.27: Yahoo! Weather forecasting service displayed in Internet Explorer [http://weather.yahoo.com]

8.7 Evaluation of the framework

The goal of the evaluation of the MITL framework is not the measurement of the internal/external quality of the software product but the estimation of quality in use – the product's acceptance by different users in particular contexts. The evaluation takes into consideration the ISO 9126 standard and the Device Independence Authoring Challenges. Additionally, it focuses on software usability and quality in use measured with the help of Quality in Use Integrated Measurement (QUIM). All these quality standards and evaluation criteria were introduced in section 4.3 of this work.

Basing on the QUIM's usability factors and the ISO 9126 quality model a questionnaire was prepared. It is presented in Appendix 5 and consists of five parts. Two groups of users were asked to answer the questions: thirty students with basic experience in programming and ten professional developers, creating mainly Java-based software. Both groups were introduced to the programming for mobile devices (WML, XHTML, WCSS, Java ME). They were also provided with a short tutorial on MITL and its IDE. Subsequently, the users were asked to program a simple and a complex mobile application.

The developed questionnaire should help the students and developers to assess the quality and usability of the developed software for programming of device-independent applications. Its first section concerned previous programming experience of the participants, especially in the area of mobile computing. In the first group, the students were between 19 and 23 years old and had previous programming experience in HTML, VB, VBA, and Java SE. Their overall programming knowledge was weak or average, only 2 students estimated their programming skills as good. Nobody programmed wireless applications before. In the second group the situation was totally different. The developers were between 29 and 36 years old and had more than 8 years of experience in commercial software development. They developed applications in C, C++, C#.Net and Java (Java SE and Java EE). One developer has programmed in Java ME and WML before, the rest knew languages such as HTML, XHTML, or XML but had no previous experience with mobile applications.

In the second part of the questionnaire, the users were asked to provide some general evaluations of the software. Survey's contributors had also the possibility to propose some changes and improvements for the evaluated library and development environment. Part three focused on the usability and quality in use of the proposed product. The usability criteria from QUIM were tailored to the needs of the evaluation. Part four concerned the Device Independence Principles and the compliance of MITL with them. In the last part, the users should assess to which grade MITL matches the Device Independence Authoring Challenges from the W3C.

The results of the evaluation were quite satisfying. Both groups of users were of the opinion that MITL is sufficiently powerful to create simple and complex mobile applications. They stated that they would choose MITL for developing mobile interfaces rather than using a particular markup language and performing multiple-source authoring. The surveyed students confirmed that MITL can be used by non-professional programmers and occasional users because of the syntax that is similar to other commonly known markup languages. The Integrated Development Environment also collected positive judgments. Users affirmed that it facilitates the development of mobile programs and that the provided functionality is quite self-explanatory. Participants inexperienced in programming were of the opinion

that the syntax of MITL caused the most troubles during the development of applications while experienced developers had almost no problems with the XML-based structure of MITL. Table 8.20 summarizes the results of the evaluation and presents the obtained scores.

Interesting and helpful comments were given by survey's participants in question B16, where they expressed their suggestions for changes and improvements of MITL and Integrated Development Environment. Some necessary improvements became also evident during the observation of users. While developing with the help of the IDE, most students made errors professional programmers would have never thought of. For example, they entered references to non-existing cards or HTML links, forgot anchors (“#”) when using hyperlinks, omitted the number of columns for tables (a necessary attribute in WML), etc. All proposed improvements and observed deficiencies were taken into account during the process of programming of a new, enhanced version of MITL and its IDE. In the development environment the validation functionality was added. This feature reduces to a great extent the number of syntactical errors made by users without programming know-how.

The users questioned in the survey confirmed that the most important advantage of MITL is the fact that the language can be used by non-professional programmers and occasional users without previous experience with mobile applications. This is due to the fact that the syntax is similar to other, commonly known markup languages. The user does not have to know the elements of MITL since the pages can be generated using the IDE. Compared with other languages, e.g. UIML or RIML, the user is able to develop device-independent applications after a short introductory course, because of a limited number of elements implemented and transparent language structure. As the history of SGML and HTML showed, these are very important features for the wide acceptance of new solutions.

Furthermore, the users appreciated the fact that MITL accepts input from a database, XML, and text files but most of them were not familiar with Web Language and would prefer an automatic conversion of HTML to XML, without the necessity to write scripts. They were also discouraged from the use of RSS feeds and asked for a more advanced support for parsing XML files.

In the third part of the survey, the focus was put on the understandability, learnability, and operability of the MITL framework and the IDE for MITL. From the QUIM model, the efficiency, satisfaction, trustfulness, accessibility, universality, and usefulness criteria were examined. In the second group, the users had no problems with MITL, JSTL, Web Language, RSS libraries, and the concept of JSP tag libraries. Inexperienced users were able to write an application in MITL, but got a little bit confused with the concept of tag libraries and encountered some difficulties while trying to use JSTL, RSS libraries, and Web Language. Both groups of users evaluated satisfaction, trustfulness, efficiency, accessibility, universality, and usefulness of MITL and its IDE relatively high (cf. table 8.20 for a detailed overview). Discrepancies between the opinions of developers and students have to be traced back to the programming knowledge in both groups. This became particularly evident in the evaluation of JSTL, RSS libraries, and Web Language that were positively ranked by programmers and negatively by the first group of users.

Question	Average Students	Standard Deviation Students	Average Developers	Standard Deviation Developers
B1	6.8	0.4	6.3	0.48
B2	6.74	0.57	6.5	0.53
B3	6.6	0.6	6.0	0.66
B4	1.09	0.3	1.2	0.42
B5	6.4	0.5	6.4	0.51
B6	6.0	0.96	6.0	0.94
B7	6.54	0.62	6.6	0.7
B8	6.7	0.54	6.2	0.79
B9	6.58	0.56	6.4	0.7
B10	6.0	0.83	6.5	0.53
B11	1.38	0.49	1.3	0.48
B12	6.8	0.4	6.8	0.42
B13	6.22	0.92	6.6	0.7
C1	5.13	1.01	7	0.0
C2	5.6	1.1	6.5	0.53
C3	4.66	0.71	5.9	0.31
C4	3.7	1.08	5.1	0.31
C5	3.8	1.2	4.9	0.53
C6	5.6	0.77	6.2	0.63
C7	5.43	0.62	5.9	0.31
C8	1.2	0.4	1.1	0.31
C9	6.13	0.57	5.8	0.42
C10	7.9	0.76	5.9	0.73
C11	5.9	0.61	5.5	0.70
C12	5.5	0.51	5.9	0.57
C13	5.86	0.63	6.2	0.42
C14	4.96	0.96	5.0	0.81
C15	5.6	0.86	5.7	0.48
C16	5.53	0.77	6.6	0.51
C17	6.03	0.85	6.4	0.51
C18	6.16	0.79	6.6	0.52
C19	6.13	0.82	6.5	0.53
D1	5.96	0.94	6.4	0.7
D2	7.0	0	7.0	0
D3	6.2	0.73	6.5	0.53
D4	6.0	0.79	6.6	0.7
D5	6.2	0.8	5.3	1.05
Scale: 1. Strongly Disagree 2. Moderately Disagree 3. Mildly Disagree 4. Neither Disagree nor Agree 5. Mildly Agree 6. Moderately Agree 7. Strongly Agree				

Table 8.20: Evaluation of MITL and IDE (averages and standard deviations for two groups of users, cf. Appendix 5 for questions)

Approach Challenge/ Evaluation criteria	Evaluation of MITL
Provide comprehensive scope	
DIAC-3.1: Application scope	***
DIAC-3.19;3.20: Integration of device-dependent and independent content	****
DIAC-3.28: Range of complexity	****
Support smooth extensibility	
DIAC-3.2: Extensible capabilities	****
DIAC-3.29: Scalability of complexity	****
Support simplicity	
DIAC-3.4: Simplicity	****
DIAC-3.11: Simple content	***
Support delivery context variability	
DIAC-3.5: Navigation variability	***
DIAC-3.6: Organization variability	***
DIAC-3.7: Media variability	**
Support author specified variability	
DIAC-3.12: Text content variety	****
DIAC-3.13: Media resource variety	**
DIAC-3.14;3.15: Media resource specification & selection	**
DIAC-3.30;3.31: Aggregation & decomposition	***
DIAC-4.3: Layout variety	***
Affordability	
DIAC-3.3: Affordability (cost)	****
DIAC-3.33: Reusing existing applications	****
DIAC-6.5: Minimization of effort	****
DIAC-6.13: Separation of device-dependent and device-independent material	***
DIAC-6.6: Abstraction of device knowledge	****
DIAC-6.10: Scalability of effort/quality	****
Scale: *: non existing; **: low; ***: middle; ****: high; -: does not apply to this method	

Table 8.21: Evaluation of MITL with the help of Device Independence Authoring Challenges (averages computed from users' estimations)

With regard to Device Independence Authoring Challenges, the users rated media and delivery context variability as the biggest weakness of the library. The library supports the variability of text and images, but does not offer any support for different media types such as video or audio. Furthermore, explicit layout schemes are not available. The developer has to use tables and the inclusion/exclusion technique (`MobileYesTag`, etc.) for this purpose. Since most of the layouts are based on tables, this can lead to a better and easier applicability of the framework. An alternative solution could represent a complete separation between presentation and content elements. Such separation would enable rapid

aggregation and defragmentation of presentation units, but could result in lower acceptance of inexperienced users.

The framework was evaluated as extensible and scalable. The scalability of effort is easy to achieve and authors can influence the time and resources invested in the application development. The affordability of the framework was assessed as high because everybody can download MITL and its IDE at no cost. The results of the evaluation (from the DIAC perspective) are summarized in table 8.21.

9

Mobile Web Services Adaptation framework

The main force behind software development is the need to confront rapid change. Programs can no longer be built from scratch; they have to be assembled from existing code, components, frameworks, and applications. Strong competition, the need to increase enterprise business agility in response to new market requirements, and the growing number of not integrated legacy systems forces enterprises to search for new architectural solutions for their IT systems. According to the Lehman's law, applications should be able to adapt to the environmental evolution [Lehm80]. This adaptation really occurred: while in the nineties the focus was mainly put on enclosing system functionalities into layers and their assignment to different tiers in a system, the current decade is dominated by the idea of Service-Oriented Architectures (SOAs) and their specific implementation in the form of Web Services.

This new flavor of distributed computing is considered as a very promising technology for common software integration problems. Web Services are self-contained, standard-based, network-accessible interfaces describing a collection of operations [W3C04h]. They enable interoperability between various programs running on heterogeneous platforms within one company or across many enterprises. Web Services can easily be added to existing systems because they provide a sort of supplementary wrapper to the relevant functionalities.

The rapid growth of the Internet, mobile networks and services happened simultaneously and were not correlated due to the fragmentation of frameworks and interfaces, disparate technology standards, high integration costs, and different business models. It is a difficult and challenging task to provide Internet services to mobile applications, and to integrate mobile network services with computer-based applications. Mobile Web Services can serve as a bridge that enables the convergence and integration of mobile and Internet standards. Web Services are platform-independent and can easily cope with the heterogeneity of mobile platforms. They are located on remote servers and can offer permanent access to crucial business functionalities at anytime and from anywhere. Moreover, a mobile application can be composed from many existing Web Services.

Mobile applications can benefit from implemented Web Services and could theoretically exhibit their own functionalities as Web Services. Such services are rather limited in scope due to the restricted capabilities of mobile devices, particularly instable connections and limited processing capabilities. Mobile phones can host services that may be useful to other services on the network. For example, a Web Service located on a mobile device can provide information about the device location and preferences of the user of the device.

This chapter describes a solution that enables communication between Web Services and fat or thin mobile clients. Section 9.1 provides an overview of the evolution of architectural solutions based on the paradigm of component orientation and introduces the concept of Service-Oriented Architectures (SOAs). In the next section, Web Services and their underlying technologies are outlined. Section 9.3 describes the current state of the art in the area of mobile Web Services. The subsequent section explains the design objectives of the solution. Section 9.5 describes the architecture of the Mobile Web Services Adaptation (MoWeSA) framework. The next section shortly introduces new elements in the Integrated Development Environment that was previously developed for MITL. Section 9.7 presents some sample applications developed with the help of the MoWeSA library. In the last section, the evaluation of the framework is provided.

9.1 From components to Service-Oriented Architectures

In the past few decades, software architectures underwent a significant evolution from distributed systems to Service-Oriented Architectures (SOAs) [cf. Albi03; Voge03]. With the emergence of distributed systems in the 80s and 90s, the paradigm of component orientation became increasingly popular. A distributed system is defined as a collection of autonomous computers linked by a network that appears to its users as a single computer [TaSt02]. The concept of component-based architectures refers to the ideas of modularization and information hiding. Component-based architectures enable the assembly of software systems from building blocks called software components.

One of the most popular definitions states that a software component is “a unit of composition with contractually specified interfaces and explicit context dependencies only (...) that can be deployed independently and is subject to composition by third parties” [Szyp98]. The most important properties of a component are: specified, parametrizable interfaces, explicit dependencies, independent deployment, and third-party composition. Components can be used as “black boxes”, without knowledge about their internals. All of the interactions between a component and a software system using it are done with the help of interfaces.

In the component-based development, definition languages for interfaces or other resources are added to enforce the conformance of all components to the component interface, the platform and/or language independence, support for distribution, etc. The component model is standardized and defines the properties of a software component down to the technical details about the interfaces specification and component deployment. Examples of component models include the Component Object Model (COM) from Microsoft [Micr05], Enterprise Java Beans (EJB) from Sun [Roma+05] and the CORBA Component Model (CCM) from OMG [OMG04].

The developers of components try to extract the commonalities of future systems so that the developed components could be used as prefabricated building blocks. It is, however, hard to cope with possible changes that are unknown at the design time and to maintain a system without access to the internals of components. Since the use of parameters is the main technique for modifying components, changeable parameters have to be foreseen by component developers.

The main advantages of a component-based architecture are its reusability and the possibility of re-purposing. Service-Oriented Architecture that is considered as a successor of the component-based architecture offers similar benefits. The evolution of component-based architectures towards Service-Oriented Architectures was mainly brought by the decreasing cost of reliable, high-bandwidth Internet connections and the shortcomings of component-oriented technologies such as DCOM [Redm97] or CORBA. The development of distributed, component-based systems using these technologies was problematic when the participating nodes were located across the Internet (problems with firewalls) or heterogeneous platforms were used (e.g. COM on Windows interfacing with CORBA on Unix). Service-Oriented Architectures and their particular implementation in the form of Web Services eliminate these problems.

Service-Oriented Architectures (SOAs)

Information technologies are moving towards a new stage, where applications will be dynamically deployed, installed, updated, and reconfigured. Service-Oriented Architectures are an important step in this process and enjoy growing recognition and acceptance in the industry. In 2003, Gartner predicted in its report "Service-Oriented Architecture: Mainstream Straight Ahead " that "by 2008, SOA will be the prevailing software engineering practice, ending the 40-year domination of monolithic software architecture (0.7 probability)." [Puli03]. A March 2005 survey of 100 CIOs, undertaken by Smith Barney, confirmed the fact that the development of Service-Oriented Architectures is CIOs' number one priority in the area of emerging technologies [WeMe05, p.3].

SOA is an architectural approach, according to which applications are assembled from reusable components ("services") that can be published, discovered, and invoked over a network. This model provides the infrastructure for organizations' systems to dynamically react to changing business conditions, while the technical details of the implementation remain transparent. A service is a software building block that performs a distinct function through a well-defined interface. Services in SOA should be [McGo+03; W3C04g]:

- dynamic and discoverable - services can be discovered with the help of a directory (registry) and can be bound dynamically at runtime,
- modular - a service represents a discrete unit of business, application, or system functionality but possible dependencies between services are limited. A service contains furthermore a set of interfaces that should be cohesive, i.e., related to each other in the context of a module.
- coarse-grained - the granularity refers to the functionality scope that a service exposes. Fine-grained services are not very useful for business processes, while coarse-grained services encompass complete functionalities that are relevant for business processes.
- location-transparent - the architecture is unaware of the location of services, or at least, independent from it.
- loosely coupled, diversely owned - SOAs are composed of multiple services owned by different providers. These services are treated as blackbox components by service consumers since they cannot access their internals (the "code behind"). Services should have a few well-known dependencies and should not be replaced by other services.

- interoperable, network-addressable, and composable - SOAs can be created from multiple existing services using predefined standards that guarantee their interoperability. They can be developed for different platforms using various programming languages and invoked across the network.
- self-healing - services can recover from errors without human intervention, they can be rediscovered and rebound after they fail.

In SOA, service interfaces are separated from service implementation (cf. figure 9.1). A service interface (the so-called service description) explains how to call a service by specifying its location and the format of input/output parameters. It provides necessary information for making a request to the service and getting a response. The implementation of a service (the code) is responsible for the functionality (business logic) of the service. Service implementations are platform-dependent.

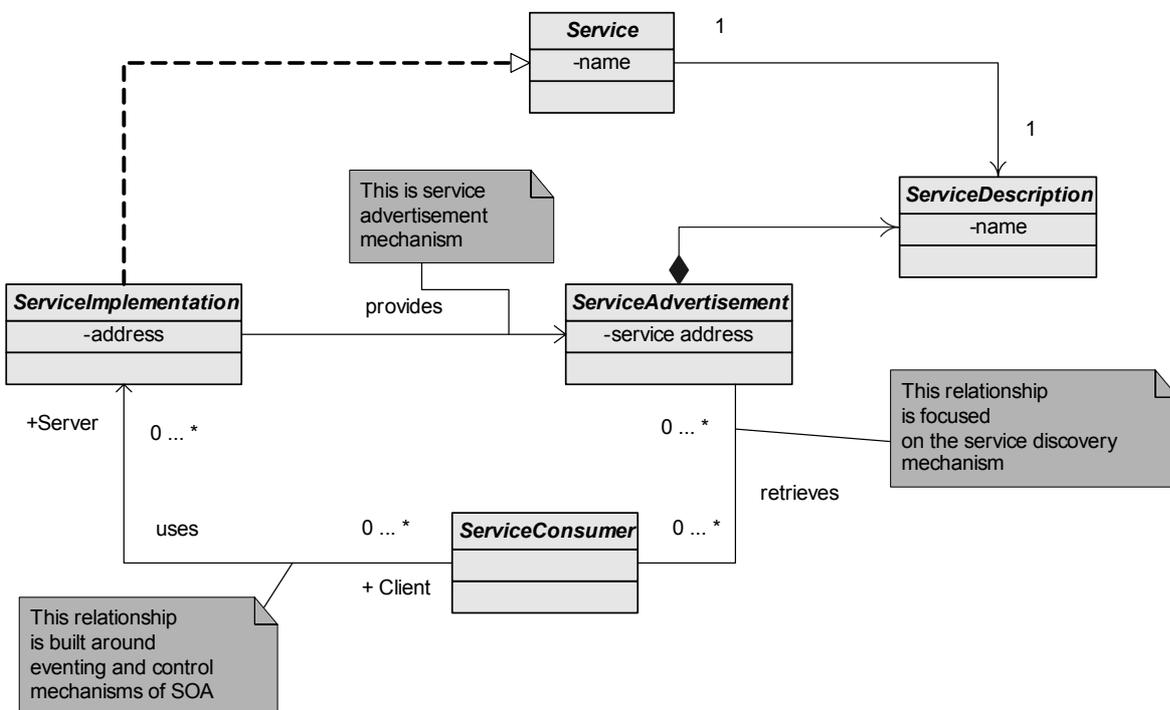


Figure 9.1: UML description of SOA [McGo+03, pp. 37, 39]

In SOA, six entity types can be distinguished: service consumer, service provider, service registry (also called broker), service contract, service proxy, and service lease [cf. McGo+03, pp. 35-38]. The service consumer is some kind of a software module (e.g. application, another service, etc.) that requires the service. It works according to the "find, bind, and execute" paradigm. The service consumer locates the required service in a registry, binds to it over a transport mechanism, and executes its functions. The service provider is an entity that publishes its contract in a registry, accepts and executes the requests of a customer. A service registry is a network-based directory that stores contracts from providers and displays them to customers. In a service contract, the interactions between service provider and consumer are specified. It discloses the format of requests and responses and may also indicate pre- and post-conditions for service execution. Alternatively, it may contain information about the nonfunctional aspects of a service such as the quality of service (QoS) levels. A service lease limits the

amount of time for which a contract is valid. The concept of service lease does not possess any implementation details so far. Service proxy is a supplementary piece of software that helps to access the services. It finds a contract in a registry, formats the messages, and executes requests according to a service description. It can also cache remote references and data.

Service-Oriented Architectures offer different benefits. They help to reduce time to market (TTM) for new services and decrease the total cost of ownership (TCO) of IT infrastructure and business services. SOA eases the integration of heterogeneous environments found in many organizations through the use of standard protocols. Legacy systems can be easily integrated and chunks of SOA-based components can be reused in different applications. Loose coupling increases organizational agility by allowing companies to assemble and modify business processes in response to changing market requirements. Due to the modular nature of SOA, incremental development, deployment, and maintenance are possible. The existing investments in IT assets can be leveraged since SOA wraps the available business functionalities; risk and development efforts are therefore lowered. This is particularly important for small enterprises that consider each investment carefully and are not eager to apply new solutions if old applications are still working. Companies may use software and hardware of their choice and can engage in a multi-source strategy, reducing the threat of vendor lock-in.

In the book "Patterns of Enterprise Application Architecture", Martin Fowler states that loosely-coupled, distributed system architectures can easily become "An architect's dream and a developer's nightmare" [Fowl03]. Service-Oriented Architectures add a new level of complexity. Configuration and deployment are more difficult, network interruptions or version mismatches between components have to be handled. Furthermore, the existing implementations often reflect the organizational structure of an enterprise rather than its business functions or processes. Simple wrapping of available applications in Web Services is not possible as it will result in redundant code and inefficiencies.

SOA should not be implemented by enterprises solely due to the fact that the IT world considers it as a revolutionary solution. A company planning to use SOA must take into account its business needs. SOA is particularly helpful if an organization has to deliver a cross-functional solution or wants to reuse its existing IT assets. SOA can be applied to systems that are aimed at the support of various technologies. Alternatively, SOA can be used if the implementation should not be disclosed to external or internal customers. Investment costs, developer's skills, and available techniques and tools have to be considered as well before starting a SOA implementation. The reliability of internal and external network connections becomes paramount in any distributed environment. In a highly interconnected Service-Oriented Architecture, even a partial network failure wreaks havoc on a business.

9.2 Basic technologies for Web Services

Web Services represent the most important approach to realize Service-Oriented Architectures. A Web Service is described by the World Wide Web Consortium (W3C) as follows [W3C04h]:

"A software system designed to support interoperable machine-to-machine interaction over network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages,

typically conveyed using HTTP with XML serialization in conjunction with other Web-related standards.”

Web Services possess the most important characteristics of SOA services: they are loosely-coupled, encapsulated components that make their interfaces available through standard protocols and can be invoked over the Internet. A Web Service can represent an individual application or a modular subcomponent of a larger program. The underlying technology and the business logic of a Web Service are invisible to the users of the service. Web Services are independent from each other; they can run on different platforms and can be replaced by services with similar business functionality but implemented for different environments.

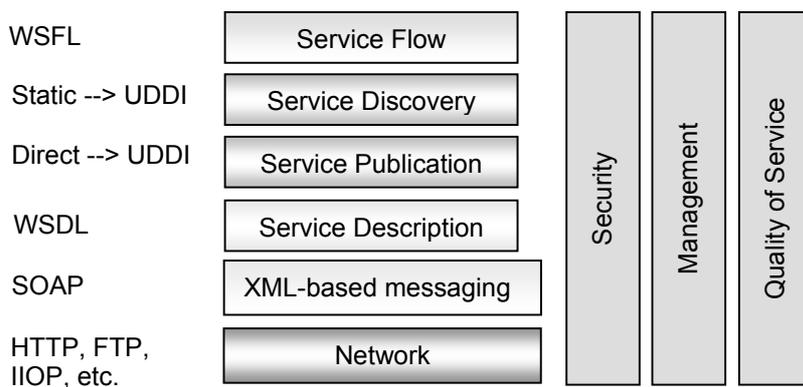


Figure 9.2: Web Services programming stack [Gott+02, p. 171]

Web Services programming stack, illustrated in figure 9.2, is a collection of standardized protocols and application programming interfaces. They allow users to allocate and use Web Services. The foundation layer of Web services is the network layer. It is usually the HTTP protocol, but alternative network protocols such as the Internet Inter-ORB Protocol (IIOP) may also be used. The communication between a user and a Web Service is performed with the help of a messaging layer based on the Simple Object Access Protocol (SOAP). The description of Web Services is enclosed in Web Services Description Language (WSDL) documents. Web Services may be published by sending an e-mail with a WSDL file directly to a client or by advertising (publishing) them in a UDDI registry [cf. Vino02]. The service flow layer facilitates the composition of Web Services into workflows and the representation of this aggregation as a higher-level Web Service. Each layer of the stack has to address the security, management, and quality of a service. Two interaction patterns are possible for Web Services: an RPC-based or a document-based interaction [cf. Vino02a].

In the RPC-based interaction, a service consumer perceives a Web Service as a single logical application or a component with encapsulated data. The exchanged messages map directly into input and output parameters of procedure calls or operations. This mapping can be defined in a WSDL document, from which the user can generate the stub code. The RPC protocol used has to specify the methods for transporting typed values between the representation in SOAP and an application's representation (e.g. a class in a programming language), and identify various RPC parts (object identity, operation name, and parameters). The RPC-style can be used only for synchronous

messaging, because a request always waits for a response. This means that RPC-style Web Services are tightly coupled, similarly to traditional distributed object paradigms like RMI.

In the document-based interaction, the user communicates with a Web Service with the help of documents (usually in the XML format) that are processed as complete entities. This type of Web Services is loosely coupled and may be used for asynchronous communication. The main differences between the RPC-based and document-based interactions are summarized in table 9.1.

RPC Style	Document Style
Emulate function calls	Exchange business documents
Payload is interpreted with schema encoding rules	XML payload is defined by schemas but it can also carry binary data
Fine-grained, tightly coupled services	Coarse-grained, loosely coupled services
Synchronous programming style model (blocking call)	Asynchronous messaging

Table 9.1: Comparison between RPC and document style for Web Services

RPC-based and document-based interactions should not be confused with the binding styles that specifies how the elements in the SOAP body are constructed. Two binding styles can be distinguished: *rpc* or *document*. In the *rpc* style extra elements are added to simulate a method call and in *document* style XML is sent (cf. the next section) [Gitm03].

Binding styles do not match with the interaction styles. It is, therefore, possible to use a *document*-style Web Service for *RPC* communication and if a Web Service has the *rpc* binding it still can be used for *document*-based interactions. Data should not be confused with communication patterns, even if the names applied may cause this confusion.

The last type of Web Services are *REST* Web Services that are rarely present in the literature but are gaining increasing popularity in practice [Goth04]. *REST* stands for the Representational State Transfer and is actually an architectural style. It is defined as “a coordinated set of architectural constraints that attempts to minimize latency and network communication, while at the same time maximizing the independence and scalability of component implementations” [FiTa02]. A *REST* Web Service is based on the concept of a resource – any information that can be characterized by a Uniform Resource Identifier (URI). In *REST* Web Services, the interfaces are limited to the HTTP protocol. They support only the create, retrieve, update, and delete (*CRUD*) methods. HTTP provides application-level semantics via GET, POST, PUT, and DELETE. HTTP GET is used by service consumers to obtain a resource's representation from a URI. HTTP DELETE is used to remove a resource's representation, HTTP POST helps to update or create a resource's representation, and HTTP PUT is used to create a resource's representation. An implementation of the *CRUD* methods is realized in the following way: a Web Service's method name and its parameters are attached to a URL, and HTTP GET is used to retrieve the data in the XML format.

Simple Object Access Protocol (SOAP)

The Simple Object Access Protocol (SOAP) standard [W3C03d; Curb+02] originates from the XML-RPC (XML Remote Procedure Calling) specification. XML-RPC is a simple protocol that enables software running on various platforms to make remote procedure (RPC) calls over the Internet. A remote procedure call is a request to an application located on a server to perform specific operations and to send the results back to the client. The specified remote server executes the call and returns the data in the XML format [Wine99]. SOAP is similar to XML-RPC but its parameters are notational (a request must have the method parameter names encoded within the XML) rather than positional (parameters recognized by position).

SOAP specifies a method to wrap up information so that it can be transmitted between peers. These peers are able to interpret the information and respond to it. SOAP is designed to exchange messages containing structured and typed data. It does not specify a set of XML elements and attributes for primitive data types (for example integers or strings). It prescribes, however, an encoding mechanism for these data types. SOAP data types can be divided into two broad categories: scalar types and compound types [cf. Cera02, pp. 48-52]. Scalar types contain only one value (e.g. a stock price) whereas compound types contain multiple values (e.g. a list of stock quotes). Compound types are further subdivided into arrays and structs. In arrays the elements are specified by their position (index), in structs they can be accessed by an accessor name. For scalars, SOAP adopts all the built-in data types specified in the XML Schema specification. This includes boolean, String, float, short, long, date, time, double, and integer. For arrays in SOAP, the developer has to specify the element type and array size. Depending on the implementation, SOAP can also support multidimensional arrays.

SOAP messages are available in two flavors: as RPC-style and document-style messages [cf. Gitm03]. Listing 9.1 and 9.2 show examples of both types, respectively. In the RPC-style the root element is qualified with a namespace and the child elements are unqualified names. The name of the root element comes from the operation's name for a SOAP request and in a SOAP response the word "Response" is added to the operation's name. The children elements represent parameters of the operations or returned values. In document-style all elements are defined within the same namespace. The XML elements are explicitly defined.

```
1. <tns:matchNoteAndNote xmlns:tns="urn:notes.demo">
2.   <in0 xsi:type="xsd:string">0000000000</in0>
3.   <in1 xsi:type="xsd:string">000000000B</in1>
4. </tns:matchNoteAndNote>
```

Listing 9.1: Example of RPC-style message

```
1. <out:getNoteResponse xmlns:out="urn:notes.demo">
2.   <out:note key="000000000B" >
3.     <out:content>test</out:content>
4.   </out:note>
5. </out:getNoteResponse>
```

Listing 9.2: Example of document-style message

SOAP can furthermore be encoded as a literal or SOAP-encoded message [Gitm03]. Usually, literal encoding is used with document-style Web Services and SOAP encoding is applied to the RPC binding style. In literal encoding, a SOAP message has to follow an XML Schema definition. SOAP encoding uses a set of rules defined in the SOAP specification. The rules are based on the XML Schema data types for data encoding, but no particular schema is specified. Literal encoding enables validation of messages and transformation with XSLT, SOAP-encoded messages do not offer such possibilities.

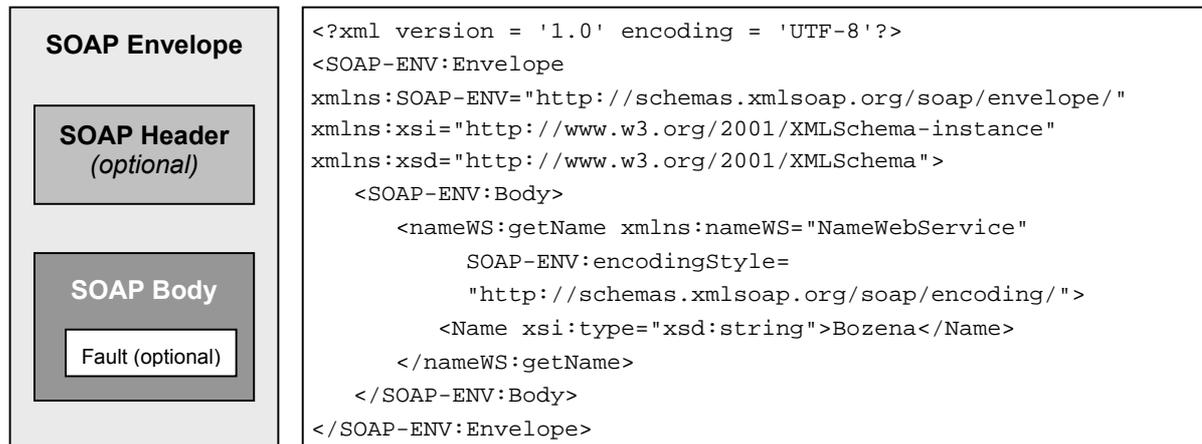


Figure 9.3: Structure of SOAP and a sample SOAP response

SOAP can be used on top of several transfer protocols like HTTP, SMTP (Simple Mail Transfer Protocol), and FTP (File Transfer Protocol). It does not define any application semantics; only the basic structure of SOAP messages was specified. A SOAP message consists of three parts belonging to a namespace defined by the SOAP specification: an envelope, an optional header, and a mandatory body (cf. figure 9.3). The envelope element is the root element of a message. The header element can carry additional information useful in the processing or routing of the message payload. Digital signatures to guarantee the integrity of data or authentication information to validate the identity of the message sender are typical examples of header data. The body element contains the payload of a message in the XML format. The “SOAP with Attachments” specification [W3C00] allows a SOAP message to have associated MIME attachments that can contain different types of data. Attachments usually include images, audio, or plain text. Each attached object is added as MIME content, and the whole message is packaged as a MIME Multipart/Related message as defined by RFC 2387 [Levi98].

Web Services Description Language (WSDL)

The Web Services Description Language (WSDL) standard [W3C01a] provides a description of an abstract interface through which a service consumer communicates with a service provider. It also specifies the details of a particular implementation of this interface [Curb+02, pp. 88-90]. In WSDL files, four types of information are defined: data types, messages, interfaces, and services (cf. figure 9.4).

A service describes a specific network location and consists of a collection of ports. A port has an abstract definition (the so-called port type) and a concrete definition (a binding). Web service interfaces are similar to the interfaces defined in object-oriented languages and are called port types. They include input messages (the set of parameters passed into an operation), output messages (the set of

values returned by an operation), and fault messages (the set of error messages delivered if an error occurs). Binding specifies how an interface is bound to specific transport and messaging protocols (such as SOAP and HTTP). A message is a logical collection of named parts (data values) of a particular type. The data types of parts are defined by a service provider using XML Schema.

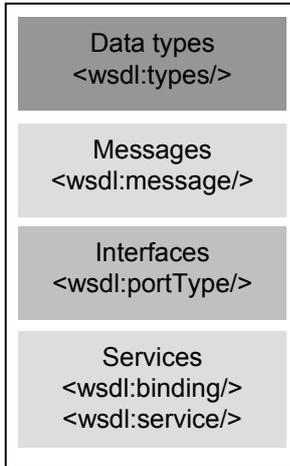


Figure 9.4: WSDL structure

Universal Description, Discovery and Integration (UDDI)

The Universal Description, Discovery and Integration (UDDI) [OASI03] specification helps service consumers to find service providers through a centralized registry. UDDI consists of two parts: a registry of all Web Services metadata (including a pointer to the WSDL description of a service), and a set of WSDL port type definitions for manipulating and searching the registry [Curb+02, pp. 90-92].

UDDI registries are databases containing three types of pages with information about Web Services:

- “white pages” with information about names and contact details of service providers
- “yellow pages” information providing a categorization based on business and service types with the help of standard taxonomies
- “green pages” containing technical data about the services.

Registry access is accomplished using a standard SOAP API for both querying and updating the information about Web Services. Public UDDI registries were shut down by their providers such as IBM, Microsoft, SAP, Systinet, or Oracle in January 2006 and are no longer accessible¹³³. Private registries (inter- and intra-enterprise registries), developed and used for internal purposes, and the support for UDDI entries in products of different providers¹³⁴ are supported.

¹³³ Cf. http://www-306.ibm.com/software/solutions/webservices/uddi/shutdown_faq.html.

¹³⁴ Cf. <http://uddi.org/solutions.html>.

Four main data structures can be identified in the UDDI specification: `businessEntity`, `businessService`, `bindingTemplate`, and `tModel`. The `businessEntity` data structure contains information about the provider of a Web service (“white pages”). The `businessService` structure consists of information about a particular service and contains one or more `bindingTemplates` indicating binding information of the service. One of the most important parts of the `bindingTemplate` structure is the access point of a service (e.g. an URL, telephone number, etc.). A `bindingTemplate` may possess several references to `tModels`. A `tModel` (“green pages”) may contain a link to a WSDL document or may include a taxonomy of classification systems used for the categorization of offered Web Services. Three standard taxonomies are applied in the UDDI registries:

- an industry classification in compliance with the North American Industry Classification System (NAICS) taxonomy
- a classification of products and services complying with the Universal Standard Products and Services Code System (UNSPSC) taxonomy
- a geographical categorization system complying with geographic taxonomy of the International Organization for Standardization (ISO 3166).

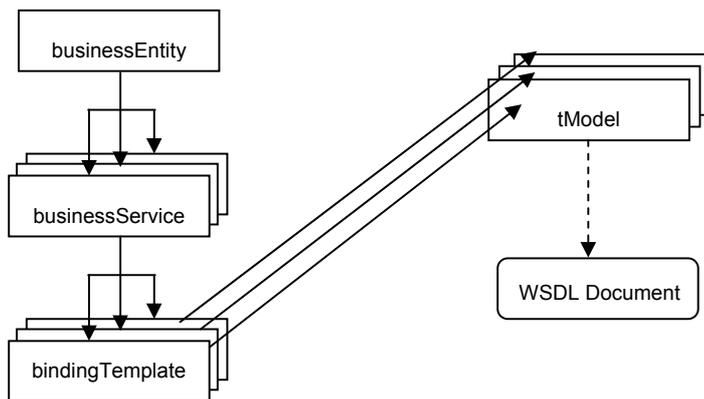


Figure 9.5: Basic UDDI data structures

Using RPC-based and document-based Web Services

The main steps in the RPC-based Web Services paradigm are as follows (cf. figure 9.6) [Vino02a]:

- 1) A developer writes an application that is conform with Web Services standards
- 2) The application is deployed to become available to other applications (it is, for example, packaged into a Java EE WAR file and deployed to a Java EE Web container, e.g. Tomcat)
- 3) An abstract service description in the form of a WSDL file is provided by the service developer
- 4) The service is published in the UDDI registry. The registration process includes three steps: posting the information about the creators of the service, categorization of the service, and making the URL of the WSDL file available to customers
- 5) A client developer searches the UDDI registry for a service that fulfills specific requirements and retrieves the appropriate WSDL file

- 6) The WSDL file is used as an input to a stub generator program. This program automatically produces the code that is necessary to make calls to a remote service
- 7) The developer incorporates the stub code in his/her application and makes calls to it. The stub is responsible for SOAP messaging, HTTP binding, and RPC connectivity. The stub is also in charge of handling service responses.

The document-style Web Services are characterized by a higher degree of complexity. Therefore step 7 is different. A programmer has to create an outgoing XML document and has to parse the incoming data.

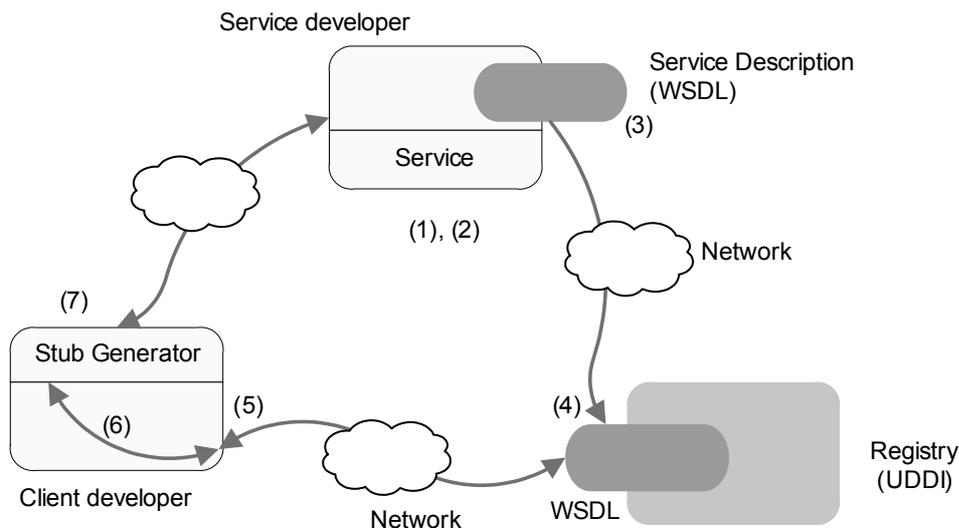


Figure 9.6: Main steps in the Web Services paradigm [own illustration]

The communication between a client and a Web Service is a complicated process. For RPC-based interactions, the transformation to XML and back is possible with the help of two processes – marshaling/unmarshaling and serialization/deserialization. Figure 9.7 illustrates the communication mechanism in an RPC-based Web Service. The client first calls a method implemented as a remote procedure. A surrogate for this procedure represents a proxy stub that acts as an external interface to the caller and implements the processes required for the preparation and transportation of data.

The stub collects the received list of parameters into a SOAP message. This is possible with the so-called marshaling. The stub encodes the parameters to make sure that the Web Service will be able to interpret them correctly. Encoding can consist of the identification of the SOAP structure and data types or conversion of data to correct types. Subsequently, the stub serializes the generated SOAP message across the transport layer to the server, on which the Web Service runs. Serialization includes the process of converting the SOAP message into a TCP/IP buffer stream and transporting this stream to the server.

On the server, the reverse process takes place. A listener service detects the data, deserializes them and calls a stub on the server. The stub (often called a tie [Fost+02, p.305]) unmarshals the parameters, decodes and binds them to internal variables/data structures. In the last step, the stub

invokes the called procedure. Then the server and client change the roles and the server performs marshaling and serialization to return the data to the client [cf. Holl03].

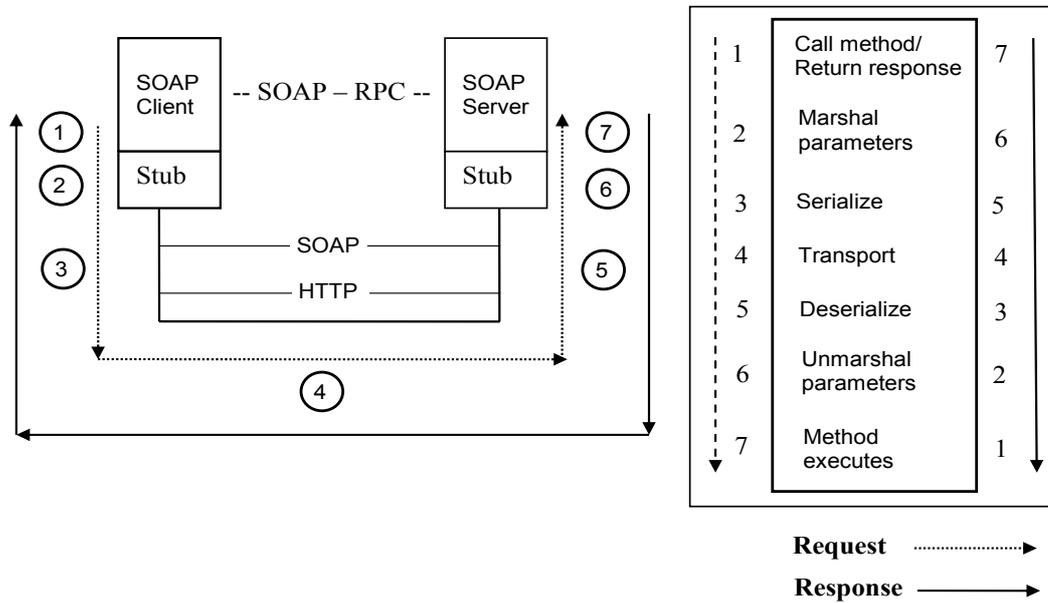


Figure 9.7: Communication process in an RPC-based Web Service [Holl03]

In the document-style message exchange, illustrated in figure 9.8, an XML document is first created on the client with the help of XSLT and some parser (e.g. a DOM- or SAX-based XML parser). The generated XML is placed in the <Body> element of a SOAP message. The client can include in the message a reference to a namespace. This reference is not obligatory and helps applications to validate the content and format of the XML document. If the namespace is missing, the client and server have to agree on some schema for the validation and interpretation of XML messages.

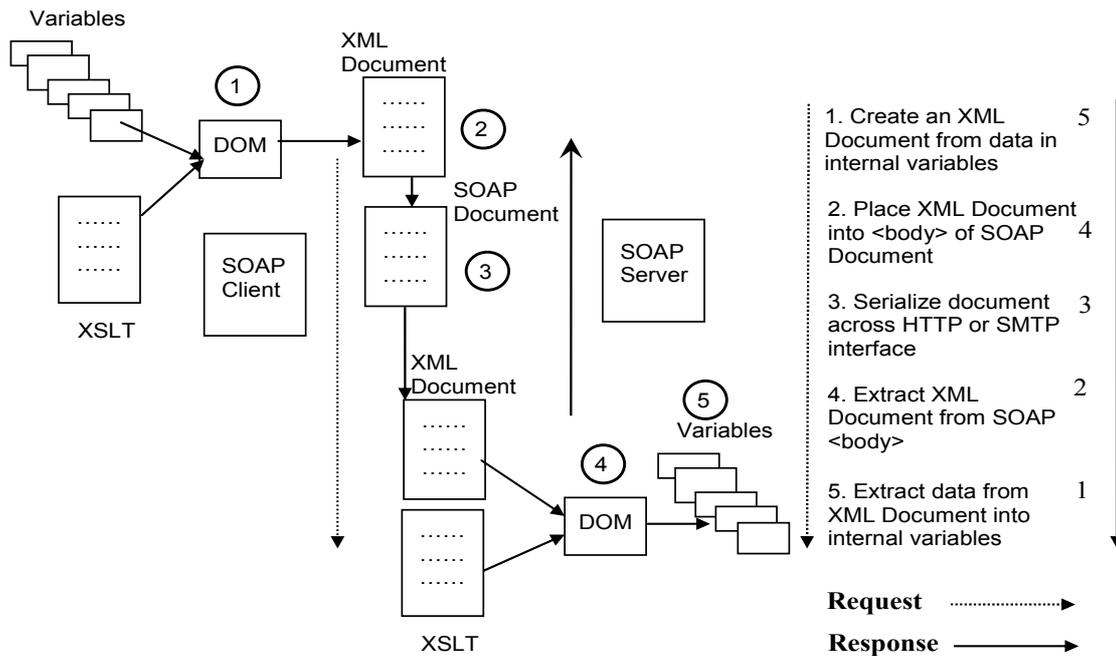


Figure 9.8: Communication process in a document-based Web Service [Holl03]

The SOAP client serializes the message and sends it to the server that reverses the described process. The server validates, extracts, and binds the information from the XML document to its own internal variables. Subsequently, the server has to produce XML that is sent to the client as a response.

RPC-style messaging is a standards-based, platform-independent component technology. Clients and servers can apply different programming languages to develop the respective interfaces. RPC-based Web Services are strongly coupled: changes in the number, order or data types of the parameters require changes in interfaces on the client and server side. RPC calls are synchronous in their nature; it is possible to make them asynchronous but this adds additional complexity to the application. Marshaling and serialization of XML generates some overhead in comparison to binary data streams, but it is not a problem for powerful computers.

Document-style messaging can be applied if XML documents are part of the interface parameters. This situation often occurs in systems that have to pass complex business documents, such as invoices, receipts, or customer orders. The content and structure of XML documents can be generated from original XML data with the help of an XSLT document. Changes in the structure of XML documents do not influence the interface as such because they are reflected in the XSLT style sheet. This type of interaction can be applied in both synchronous and asynchronous interfaces.

In document-based messaging, the client and server have to agree on a service identification mechanism (this information may be, for example, part of the SOAP header). Since in this interaction type XML is used, parsers have to be used to extract the data values from the messages and some sort of a Document Object Model has to be applied to create a message. In comparison with the RPC-based mechanism the usage of XML documents requires more work from a developer.

9.3 Mobile Web Services

There are two interaction possibilities between a Web Service and a mobile device. A handset can communicate with a Web Service directly if it has sufficient resources and appropriate libraries installed on it, or indirectly, with the help of a middleware that is in charge of the communication between a mobile device and a Web Service.

With the introduction of the Sun's Web Services implementation for Java ME [Sun05], developers gained integrated Java ME support for interacting with Web Services. Before the Java Specification Request 172 and its implementation were available, direct mobile access to Web Services was possible only with the help of third-party libraries.

Programmers used small Java XML parsers, such as XParse [Clas02] or ASXMLP [Sutt02] (cf. table 9.2), for generating or processing SOAP messages. Three types of parsers are available [Knud02]:

- a model parser that reads an entire document and creates its representation in memory. Such parsers need more memory because they store the whole document in it.

- a push parser that reads a document and notifies a listener object if it encounters specific parts of the document.
- a pull parser that reads only a fragment of a document at once and requests the next piece.

Name	MIDP support	Size	Type
ASXMLP [Sutt02]	Yes	6 KB	push, model
kXML 2.0 [Enhy02]	Yes	9 KB	pull
NanoXML 1.6.4 [Sche00]	No	10 KB	model
TinyXML 0.7 [Thom00]	No	12 KB	model
XParse 1.1 [Clas02]	Yes	6 KB	model
MinML 1.7 [WiPa00]	No	14 KB	push
JAXP [Sun05]	Yes	23 KB	push

Table 9.2: Existing Java XML parsers for mobile devices [based on Knud02]

The most commonly used parser, kXML, was developed by Enhydra [Enhy02]. kXML is used in kSOAP [Enhy04] and kXML-RPC [Enhy04a] libraries. kSOAP was developed to process SOAP on devices supporting Java ME. It provides its own mechanism for (de)serializing objects so that they can be sent across the network. Together with the kXML parser, this library takes up about 42 KB. kXML-RPC provides support for the XML-RPC protocol for Java ME-enabled devices and requires 24 KB of memory [GaGo02].

Another popular toolkit that is helpful in the generation of mobile Web Services is gSOAP [Enge04; Enge05]. It includes an RPC compiler supporting C and C++ Web services applications. This compiler produces code to limit the size of an application and to reduce runtime memory, network and processing requirements of Web Service applications. The gSOAP toolkit includes also an XML parser/generator and a WSDL importer. It is portable to many platforms, including powerful mobile devices.

The Java Specification Request 172 [Sun04a] was specified by the Java Community Process (JCP) in 2004. It defines a set of APIs for Web Services running in the Java ME environment. The J2ME Web Services APIs (WSA) are subsets of enterprise versions of these packages [Sun05; Sun06g]. The J2ME implementation includes the Java API for XML-based RPC (JAX-RPC) and the Java API for XML Processing (JAXP). The first library enables access to remote Web Services and the second one provides the possibility to parse XML data on mobile devices. The JAXP for J2ME supports the Simple API for XML Parsing (SAX) in version 2.0, but does not offer any support for the Document Object Model (DOM) or XSL Transformations. In the JAX-RPC for J2ME, the RPC invocation is stub-based. Dynamic proxies or dynamic invocation interfaces (DII)¹³⁵ are not supported. The client must be based

¹³⁵ DII allows dynamic creation and invocation of requests on objects whose type was unknown at the time the client was written.

at least on CLDC or CDC 1.0 and must possess 50 KB of RAM and 25 KB of ROM. The library can be used only for the document/literal binding style. Neither the support for service endpoints (Java ME device-based Web Services are not possible) nor the service discovery with the help of UDDI is possible as well.

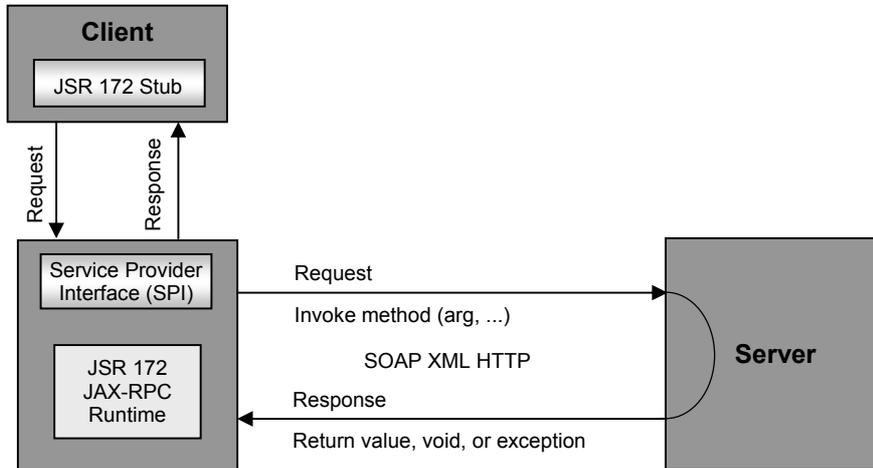


Figure 9.9: Invocation of RPC-based Web Services in Java ME [Orti04]

To interact with a Web Service from a mobile device using the JAX-RPC library, a service invocation stub has to be created. It encodes and decodes input and return values. It interfaces with the JSR 172 runtime to invoke a remote service endpoint. Stubs communicate with the runtime through the Service Provider Interface (SPI)¹³⁶. An instance of the generated stub should be created in an application's code and appropriate methods of the stub should be called to invoke a Web Service. Remote invocations from mobile handsets should be performed in separate threads of execution. This is due to the fact that remote invocations in JSR 172 are blocking, synchronous calls and the user interface will freeze until the remote invocation completes [Orti04]. Figure 9.7 illustrates the process of the invocation of RPC-based Web Services according to the JSR 172 specification.

To parse a document using the mobile version of JAXP, a developer should write an application-specific event handler that provides the actions for processing different parts of the XML document. She/he should furthermore create an instance of the SAX parser and parse the input XML document by invoking the `SAXParser.parse()` method [Orti04a]. The HTTP protocol can be used as a transport mechanism for sending requests to Web Services and obtaining SOAP responses.

Web Services can also be accessed from mobile devices with limited capabilities with the help of a browser/Java ME application and a middleware. A wireless device can send an information request specified in a supported markup language (e.g. in WML). Some kind of a Java EE-based middleware (for example JavaServer Pages residing on a Web server) translates the request to a SOAP message

¹³⁶ SPI is a mechanism used in the development of Java class libraries and standard extensions to Java. Through the use of interfaces, Java programs can invoke methods on an object whose implementation is not known at compile time. At runtime, the Java class loading mechanism is used to dynamically locate and load classes that implement the SPI [cf. SeWr02].

or an XML-RPC call and sends it to an appropriate Web Service. The response is translated back to a WML document and displayed in the mobile browser. JavaServer Pages (JSPs) can also communicate with MIDP applications via the standard HTTP protocol. Therefore, the SOAP processing can be performed on the server and only the results will be sent to the respective MIDlet.

For the interaction with Web Services, Sun offers a set of APIs and tools integrated into the Java Platform, Enterprise Edition (Java EE). The available Web Services APIs are similar to the offered Java ME APIs for Web Services and include¹³⁷:

- the Java API for XML Processing (JAXP) [Sun06a] – this library supports the processing of XML documents based on Document Object Model (DOM), Simple API for XML Parsing (SAX), and XML Style Sheet Language Transformation (XSLT).
- the Java API for XML Registries (JAXR) [Sun06b] – enables access to public and private business registries over the Web. It supports the UDDI and ebXML standards.
- Java Architecture for XML Binding (JAXB) [Sun06e] - provides a convenient method to bind an XML schema to a representation in Java code. It is therefore easy to add XML and processing functions to applications without previous knowledge of XML.
- the Java API for XML Web Services (JAX-WS) [Sun06c] (formerly Java API for XML-based RPC - JAX-RPC, the old versions of JAX-RPC can be still downloaded [Sun06d]) – is the main library in the rearchitected Web Services stack and enables client programs to make XML-based remote procedure calls (RPCs) over the Internet. With the help of this API the developer can import and export WSDL documents.
- the Java API for XML-based Messaging (JAXM) [Sun06] - is an API designed to hide the complexity of the SOAP-based communication (sending and receiving of SOAP messages). The messages can be exchanged directly between a server and a client or with the help of a messaging provider. It is possible to perform one-way messaging (called also asynchronous messaging) but only with the help of a JAXM provider. The client can send a message and subsequently perform further tasks, because a response will be sent as a separate operation in the future.
- the SOAP with Attachments API for Java (SAAJ) [Sun06f] - enables the user to generate and consume SOAP messages and SOAP messages with attachments. Similarly to JAXM, synchronous and asynchronous exchange of messages is possible.

Technical implementation details in Java

The Java API for XML Web Services (JAX-WS) is “a logical rearchitecture of Web services functionality in the open-source Java EE 5-compliant application server, released as the centerpiece of Project GlassFish.”¹³⁸ Since the implementation of the MoWeSA framework is based on Tomcat 4.31¹³⁹ and

¹³⁷ Cf. <http://java.sun.com/webservices/index.jsp>.

¹³⁸ <http://java.sun.com/webservices/jaxws/>.

J2EE 1.4, the “old” Java API for XML-based RPC (JAX-RPC) has to be used for remote procedure calls.

As was previously mentioned, the process of getting method calls and translating them into appropriate format for transmission as well as translating them back is performed by stubs and ties. Stubs/ties provide object façades for accessing Web service endpoints, i.e. they make these endpoints look like local Java objects. Stubs can be created manually by developers or automatically by utilities called JAX-RPC compilers. Usually every Java EE vendor provides its own JAX-RPC compiler (e.g. xrpcc tool from Sun). The utility reads a WSDL document and generates two Java class files from it: an endpoint interface representing the abstract portType and a stub class that implements the network protocol and a SOAP messaging style specified by the binding and port definitions. The endpoint interface defines the methods that can be invoked on the Web service endpoint. The stub class implements the endpoint interface. At runtime an instance of the generated stub class translates each method invocation into a SOAP message, and sends it to the Web service endpoint. SOAP reply messages are converted by the stub into method return values (more details in [Topl03, pp. 14-76]).

The SOAP with Attachments API for Java (SAAJ) is applied to create, read and modify SOAP messages. It consists of classes and interfaces that represent SOAP elements (such as envelope, body, etc.), XML namespaces, text nodes, and MIME attachments. SAAJ implementations may vary depending on Java EE vendor. SAAJ can be used together with JAX-RPC to send and receive messages or it can be applied with its own mechanism for messaging. SOAP messages can be built from scratch or from existing XML files [cf. Topl03, pp. 77-143].

Data types	Description
Java primitive types	boolean, byte, short, int, long, float, double
Wrapper classes for Java primitive types	Boolean, Byte, Short, Integer, Long, Float, Double
Standard Java classes	<ul style="list-style-type: none"> - java.lang.String - java.util.Calendar - java.util.Date - java.math.BigDecimal - java.math.BigInteger
Value types	Arbitrary classes that meet certain conditions can be used as method's arguments and return types.
Holder classes	Holder classes may be used as method arguments to provide a form of "pass by reference" semantics that is not directly supported by the Java programming language.
Arrays	Single- and multidimensional arrays, in which the elements are all JAX-RPC-supported types, can be used both as method arguments and for the returned value.

Table 9.3: Java data types for JAX-RPC method arguments [Topl03, p. 38]

¹³⁹ <http://tomcat.apache.org/tomcat-4.1-doc/index.html>.

JAX-RPC allows a limited range of data types that can be used as method arguments or are returned by methods as values. Table 9.3 lists data types for which the JAX-RPC specification requires support. Custom implementations may offer a built-in support for additional types such as collection classes (ArrayList, HashSet, etc.) It is also possible to use data types that are not directly supported by JAX-RPC. In this case, the developer has to write custom serializers and deserializers and reduces therefore the portability of a service [cf. also Fost+02, pp. 307-314, Mons03].

9.4 Design objectives

Web Services enable a seamless interoperation between applications or their components in a platform- and language-neutral fashion. The main goal of the MoWeSA framework is to add support for the flawless communication between Web Services and mobile thin and fat clients. The framework should support the criteria guaranteeing high software quality, in particular:

- usability, i.e. understandability, learnability, and operability – the developed markup language and IDE should be easy to understand for end-user programmers and people without extensive programming experience. Furthermore, the entire solution should be straightforward to learn and apply.
- time and resource-related efficiency - the effort required to develop device-independent applications with the help of the MoWeSA framework (in terms of time and required resources) should be much lower than the effort necessary to program own solutions for the communication between mobile devices and Web Services. Moreover, the support for a variety of devices should be possible without excessive effort.
- maintainability, i.e. changeability and analyzability – it should be easy for developers experienced in Java EE programming to analyze, change, and extend the existing solution (the markup language and the respective IDE).
- portability – the solution should be portable to different operating systems (e.g. Linux, Unix, etc.) and therefore platform-independent.

In addition to the aforementioned general objectives resulting from the requirements for software quality, the following design objectives were specified:

- broad application scope - to provide access to Web Services from mobile devices equipped with a browser supporting HTML/XHTML or WML and from handsets with the integrated Java ME support.
- interaction with Web Services - to invoke appropriate methods of Web Services, to send data to Web Services, and to deliver appropriate information to Java ME and browser-based applications with the goal to generate GUIs or browser-based graphical interfaces.
- support for all types of Web Services - to be able to communicate with RPC-style and document-style Web Services as well as with REST Web Services.
- generality and universability of the framework - similarly to MITL, the MoWeSA framework should be based on the concept of JSP tag libraries. However, the goal of this solution is to develop a

generic tag library that would be able to access any Web Service. This approach differs from the concept of custom tag libraries for specific Web Services. A custom tag library is specially designed for one service and can interact only with it [cf. Sun05b].

- simplicity - to hide the complexity of the interaction with Web Services (data transformation and parsing, calls invocation, etc.). It should be furthermore easy to develop simple applications and the complexity of development should scale smoothly with the complexity of applications.
- integration with existing solutions – it should be possible to integrate the developed framework with MITL.
- affordability, open-source solution - the development of device-independent applications should not require excessive effort or high investments, the framework should be provided as an open-source project and should be based on a technology that is available at no cost.
- support for device-dependent and device-independent content – it should be possible to include device-dependent and device-independent content in applications.
- support for the full application development cycle – it should be possible to develop applications without setting any unusual restrictions for the data representation or information flow.
- exploitation of device capabilities – it should be possible to access and take into account the delivery context. The delivery context should not be based on a proprietary solution, access to it should be possible without further limits.
- support for existing applications – the framework should provide support for including parts of existing applications during the design or at runtime.

9.5 Architecture of Mobile Web Services Adaptation framework, Web Services Tag Library

Invoking Web services directly from a JSP page or a Java application is not a good software practice. The call to a Web Service should be wrapped into a reusable component, for example a JavaBean or a tag library. The tag library represents an easy interface to use the logic of a Web service within the JSP page. It eliminates the problems of mixing Java code with JSP pages and offers the following advantages [cf. KoFr02]:

- customization and centralization of solutions for Web Services problems (e.g. unavailability of a service, obtaining bad data, etc.).
- decreased level of complexity – the combination of code for the interaction with Web Services and JSP scriptlets makes the development process of JSP pages not easy to handle. Additionally, the maintenance of such code is more difficult. A tag library for the communication with Web Services eliminates these problems and allows Web designers to create and modify JSP pages.
- decoupling Web Service calls from the JSP page – if a Web Service changes, these changes are reflected in the tag library and not in each separate JSP page that uses this Web Service.

Moreover, if the calls to Web Services are created dynamically and are not Web Service-specific, Web Services modifications do not even influence the code of the tag library.

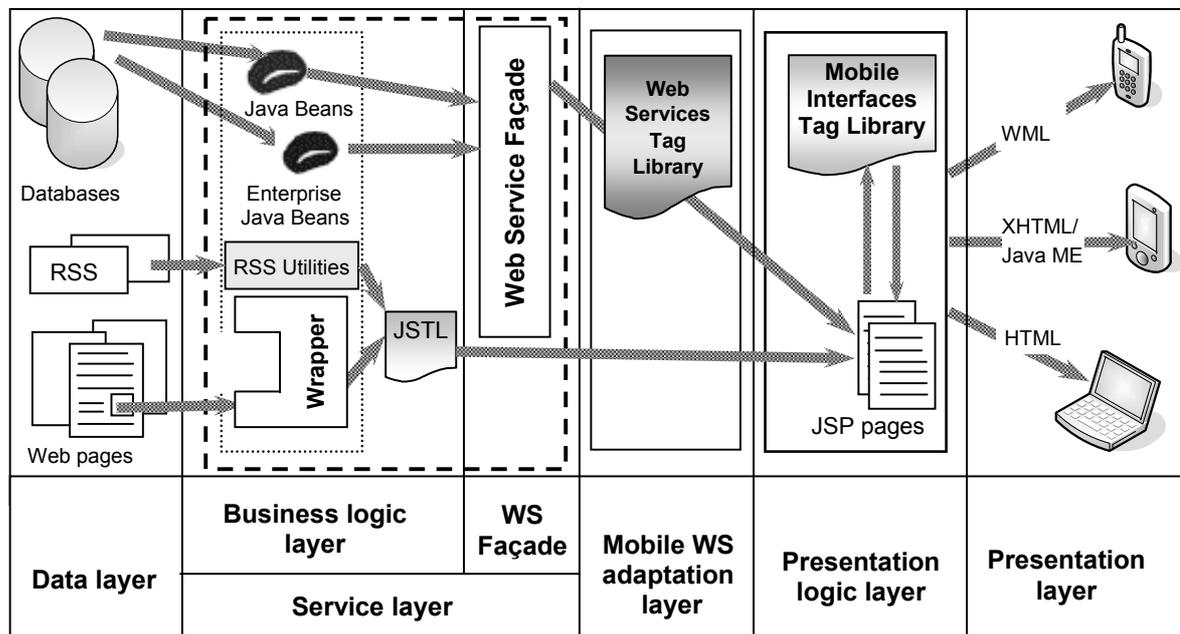


Figure 9.10: MoWeSA architecture

In order to make Web Services accessible on mobile devices, the Mobile Web Services Adaptation framework (MoWeSA) was developed. This framework uses the MITL tag library to generate mobile applications in (X)HTML, WML, and Java ME. It introduces, however, its own library for invoking RPC-based, document-based and REST Web Services. This library consists of seven tags and is called Web Services Tag Library (WSTL). Therefore, the MoWeSA framework can be considered as an extension of the MITL framework and supports the same data sources as the MITL-based adaptation approach.

The architecture of the MoWeSA framework is presented in figure 9.10. It consists of five layers. The first layer is traditionally the data layer containing different sources of information: databases, Web pages written in XHTML or HTML, RSS feeds, etc. Database access components such as Java Beans or Enterprise JavaBeans communicate with the database and deliver information to Web Services. Other sources of information are converted to XML and parsed with the help of the Web Language wrapper or RSS Utilities library. The next layer – the services layer - encompasses business logic and interfaces for this logic in the form of a Web Services Façade. The Web Services Façade uses the Façade design pattern and defines a higher-level interface for the whole system or some relevant parts of it [CaRo02, pp. 68-69]. The communication with subsystems (in this case with data access components) is performed through one, unified interface and the system is decoupled from its clients. Figure 9.11 illustrates the Façade design pattern.

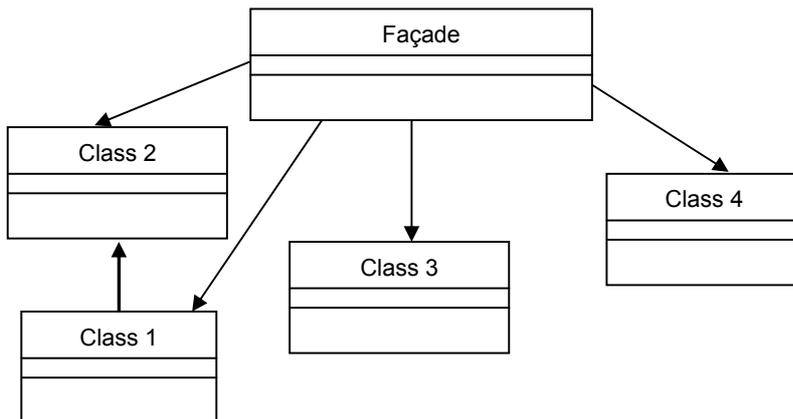


Figure 9.11: Façade Pattern

The framework can use existing Web Services that are available on the Web (e.g. Google or Amazon Web Service) or it can apply Web Services developed for a particular system/application. In the first case, the data layer and the business logic layer are completely hidden from the content author and he/she does not have to be aware of their existence. In the second case, the usage of RSS feeds and the retrieval of content from HTML pages could also be encapsulated in a Web Service.

Web Services may be in any of the available flavors: RPC-style, document-style or REST. The next layer - the mobile Web Services adaptation layer - contains the developed Web Services Tag Library (WSTL). It enables the invocation of different types of Web Services and returns SOAP responses in a format that can be processed by MITL. WSTL is based on the same technology as MITL: when a JSP engine recognizes a WSTL tag, it invokes an appropriate Java class associated with this tag. The presentation logic layer contains JavaServer Pages with MITL tags. MITL takes the results returned by the WSTL and wraps them up in appropriate tags (XHTML, HTML, and WML) or generates a Java ME application. The presentation layer consists of mobile devices (including notebooks) equipped with wireless browsers or supporting Java ME.

Web Services Tag Library (WSTL)

Web Services Tag Library (WSTL) is a generic library that can be used for the interaction with any Web Service. Specific tag libraries that target a particular solution, service, or system such as for example the Google Search service introduced in section 8.6 are still relatively scarce. Web Services Tag Library has two advantages over the custom tag library for a specific service – its syntax is general and not service-specific and developers do not have to download or develop a new tag library each time they want to invoke a new Web Service.

Web Services Tag Library allows binding to a Web Service as well as generating, sending, receiving and parsing SOAP messages. For RPC-style Web Services, it invokes methods with parameters on a Web Service and receives output. For REST Web Services, the tag library sends a HTTP GET request and interprets the obtained XML-formatted data. For document-based Web Services the tag library takes up the job of sending, receiving and parsing XML documents. Additionally, SOAP/XML can be mapped to Java data types. The WSTL library is based on the JAX-RPC and SAAJ.

The WSTL library encompasses the following tags (mandatory attributes are marked in bold type):

- **WSIdentificationTag**

syntax: `<wstl:wsit wsID="" type="doc/rpc" wsdlLocation="" endpoint="" binding="">`

The `WSIdentificationTag` creates a stub for a Web Service with a given location for a WSDL file. The stubs are generated for rpc-style and document-style Web Services from a WSDL file with the help of the `xrcc` utility and custom Java classes, developed by the author.

The document-style Web Services have, however, to satisfy the following criteria to be compatible with JAX-RPC [cf. also Top103 for more details]:

- the input and output messages have to be constructed in such a way that only one element is placed directly in the SOAP body
- the part element must use the `element` attribute instead of the `type` attribute.
- the element referenced by a port must be a `complexType` created as a sequence containing one nested element for each parameter that the operation requires.

The `wsID`, `type` and `wsdlLocation` attributes are obligatory for the `WSIdentificationTag`. The `wsID` attribute specifies the name for the stub scripting variable and references the tag. It is therefore possible to access a stub from other Web Services tags without nesting them with the `WSIdentificationTag`. The `type` attribute identifies the type of a Web Service as specified in the WSDL file. It can be either document-style (`doc`) or rpc-style (`rpc`). The `endpoint` and `binding` attributes are not obligatory and can also be retrieved from a WSDL file. The `endpoint` attribute refers to a SOAP location and the `binding` attribute corresponds to the binding information found in WSDL (specification of a protocol and data formats for the operations and messages).

- **WSOperationTag**

syntax: `<wstl:operation operationID="" operationName="" wsID="">`

The `WSOperationTag` specifies which operation (method) of a Web Service should be executed. The `operationID` attribute identifies the required operation and can be used in other tags. The `operationName` attribute provides a name of the required operation as specified in the WSDL file. The `wsID` attribute is not mandatory and can be used if the `WSOperationTag` is not nested within the `WSIdentificationTag`.

- **WSInputTag**

syntax: `<wstl:input inputID="" paramName=""/>`

The `WSInputTag` should be used if the operation requires input parameters. The `paramName` and `inputID` attributes are mandatory. The `paramName` specifies the name of a parameter, whereas the `inputID` uniquely identifies the parameter for further use.

- **WSMessageTag (only for document-style Web Services)**

syntax: <wstl:message wsID="">

The `WSMessageTag` identifies a SOAP/XML message that was created by a user and should be sent in this form as a request to a Web Service. The `wsID` attribute is not mandatory if the tag is nested in the `WSIdentificationTag`, otherwise the attribute has to be used.

- **WSMapTag**

syntax: <wstl:map name="" namespace="" javaType="" encStyle="" javaSerializer="" javaDeserializer=""/>

The `WSMapTag` provides a mapping mechanism between SOAP data types and Java types. The `name` attribute specifies the name of the SOAP data type, the `namespace` attribute provides an URI for the namespace of this data type. The `javaType` attribute specifies the data type in Java to which the SOAP type should be mapped. The `encStyle` attribute provides information about an URL for a SOAP encoding style. The `javaSerializer/javaDeserializer` attributes give the names of classes that are in charge of data serialization/deserialization. All of the attributes are obligatory.

- **WSInvokeTag**

syntax: <wstl:invoke resultID="" wsID="" operationID="" inputID=""/>

The `WSInvokeTag` invokes a specified Web Service operation with a given input and saves the result in a variable with the `resultID` name. All of the attributes are mandatory and should be specified earlier in other tags (`WSIdentificationTag`, `WSInputTag`, `WSOperationTag`). Depending on previous specification of the attributes an `rpc-style` or `document-style` service is invoked.

- **WSRestTag**

syntax: <wstl:rest URL="" returnType="parsed/xml" parserClass="" varName="" toFile=""/>

The `WSRestTag` is designed only for REST Web Services. The `URL` attribute contains the URL of the Web Service and the `returnType` attribute specifies whether the XML response should be parsed or not. If the response has to be parsed, a class that will perform this parsing should be specified. The `varName` attribute specifies the variable name in which the result of parsing should be kept for further use. The `toFile` attribute provides a name of a file to which the results in the XML format should be saved.

9.6 MoWeSA IDE

The IDE for MoWeSA is based on the IDE for MITL and extends its functionality. One additional menu with the caption “Web Services” was included into the menu bar and appropriate icons representing Web Services tags are displayed on the taskbar (as shown in figure 9.12).

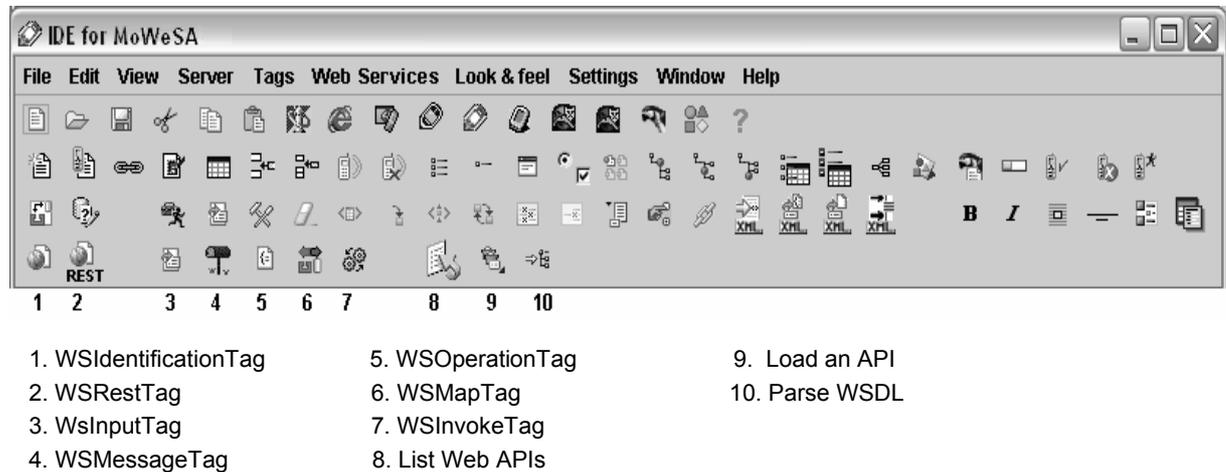


Figure 9.12: IDE for MoWeSA

The enhanced IDE is integrated with some popular Web Services and comes with the following Web APIs¹⁴⁰:

- Google Web API - supports search requests, retrieving pages from the Google cache, and spelling suggestions (<http://www.google.com/apis/index.html>)
- eBay Web API - allows to work with popular eBay features such as retrieving auctions that match specified criteria, viewing customer reviews of a seller or retrieving the status of an auction listing (<http://pages.ebay.de/entwickler/api.html>)
- PayPal Web API - offers access to secure online payment options such as retrieval of details of a particular transaction, search for transactions, sending payments, etc. (<http://developer.paypal.com>)
- MapPoint Web API - supports various localization-based services such as finding addresses, finding non-addressable places, finding nearby places, custom locations, routing, or map rendering (microsoft.com/mappoint/webservice/default.aspx).

The IDE also offers the possibility to parse a WSDL file to obtain a list with all methods offered by a particular Web Service, its parameters, required data types, binding and SOAP endpoint. Furthermore, the user can see a list of available Web APIs and can load a selected API and use it in a Web Service.

¹⁴⁰ Web APIS are publicly available application programming interfaces that can be invoked over the Internet. For more details about these Web Services and their APIs see [Gosn05; Iver04; Muel04].

If the user selects a specific API, he/she obtains intelligent, context-sensitive help that facilitates the process of filling out the attributes of WSTL tags.

9.7 Usage examples

The MoWeSA framework can be applied for document- and RPC-style Web Services as well as for REST Web Services. This section introduces two exemplary Web Services for which mobile access is provided with the help of the framework: Kayak Web Service that is implemented as XML over HTTP and a simple RPC-style Web Service that was developed from a scratch by the author of this thesis.

Kayak Web Service

Kayak Web Service offers access to the functionalities of kayak.com travel search engine. It provides three basic search possibilities: for airlines, hotels, and car rental agencies. A search has to be invoked in three steps: a developer should obtain a session, start the search, and check for results. Each search query requires therefore a sequence of the following API calls:

- 1) `GetSession` – it is necessary to obtain a new session for each search if searches are supposed to be invoked in parallel. In order to use this function, the developer has to possess a developer key.
- 2) `Start Search (for a flight or a hotel)` – this function initiates the search and returns a search id.
- 3) `Get Results (first time)` – returns a set of search results together with a search instance ID. This ID can be used in subsequent `Get Results` calls.
- 4) `Get Results (subsequent times)` – with the search instance ID attached to the URL, it is possible to inquire the Web Service every 5-10 seconds for results.

Table 9.4 presents some functions from the Kayak API and shows the structure of the URL for each of them. Since the Web Service uses REST, the results are returned in XML. Listing 9.3 displays a sample XML returned as a response of a search query for hotels in Boston. It is possible to obtain the total number of results for a given query (in this case 230) and selected information on hotels such as the current price for one night, the name and address of a hotel or the number of stars assigned to it.

Function	URL prefix with http://www.kayak.com	Parameters (name and values)	Function
GetSession		token	Provides a developer key.
		version	The version of the API (1 by default)
Start Hotel Search	/s/apisearch	basicmode	Set to "true" by default
		othercity	String locating the city. For example: "Boston, MA, US" "London, UK"
		checkin_date	In the MM/DD/YYYY format
		checkout_date	In the MM/DD/YYYY format
		guests1	Integer from 1-6
		rooms	Integer from 1-3
		apimode	Must be "1"
		sid	SessionID obtained from GetSession
		version	The version of the API (1 by default).
Get Hotel Results	/s/apibasic/hotel or /s-SEARCHINSTANCEID/apibasic/hotel	searchid	A search ID got from the start search call
		sid	SessionID got from GetSession
		c	Integer, the number of results to return
		m	Filter mode: normal or stars:? where ? is 1, 2, 3, 4 or 5
		d	Sorting direction: up or down
		s	Sorting key: price, stars, hotel, distance
		version	The version of the API (1 by default).

Table 9.4: Selected functions from Kayak Web API [<http://www.kayak.com/labs/api/search/spec.vtl>]

Listings 9.4 and 9.5 present how to invoke the Kayak Web Service with the help of WSTL and how to display the results with MITL. Listing 9.4 shows a MITL document that renders a search form with two parameters: a hotel's location and the number of stars. In listing 9.5, a sequence of calls is invoked with the help of the `WSRestTag`. The Web Service is first queried for a session ID that is necessary to perform any search. The second call to Web Services returns a search ID that is used in the subsequent invocation of a Web Service method to obtain a list of hotels matching the given criteria (in this case hotels located in Prague, with four stars). The obtained XML is stored in a file "kayakHotels.xml" and is then parsed with the "kayakHotels.xsl" stylesheet. Since 20 results are retrieved due to the value set in the "c" parameter (cf. line 14 of listing 9.5), the `PaginationTag` performs the pagination of the result set and displays the results on many pages linked by the "Next page" reference. The resulting lists of hotels in WML and XHTML are displayed in figures 9.13 and 9.14.

```

1. <?xml version="1.0" ?>
2. <searchresult>
3.   <searchid>SpHFIPitTNBu4MGtOIOs</searchid>
4.   <count>230</count>
5.   <morepending />
6.   <hotels>
7.     <hotel>
8.       <price url="/book/hotel?code=1-1_q0ld5EqKFusVVrbHAI.SpHFIPitTNBu4MGtOIOs.5-
          uTe8xW1P9WEEhLkBUrQ.H.WRIHOSTELS.2225.125734&_sid_=5-
          uTe8xW1P9WEEhLkBUrQ" currency="USD">22</price>
9.       <pricehistorylo>0.00</pricehistorylo>
10.      <pricehistoryhi>0.00</pricehistoryhi>
11.      <stars>0.0</stars>
12.      <name>YMCA of Greater Boston</name>
13.      <phone />
14.      <address>316 Huntington Avenue</address>
15.      <city>Boston</city>
16.    </hotel>
17.    <hotel>
18.      <price url="/book/hotel?code=0-1_q0ld5EqKFusVVrbHAI.SpHFIPitTNBu4MGtOIOs.5-
          uTe8xW1P9WEEhLkBUrQ.H.WRIHOSTELS.4250.125735&_sid_=5-
          uTe8xW1P9WEEhLkBUrQ" currency="USD">43</price>
19.      <pricehistorylo>0.00</pricehistorylo>
20.      <pricehistoryhi>0.00</pricehistoryhi>
21.      <stars>0.0</stars>
22.      <name>Carruth House</name>
23.      <phone />
24.      <address>30 Beaumont St</address>
25.      <city>Boston</city>
26.    </hotel>
27.    [...]
28.  </hotels>
29. </searchresult>

```

Listing 9.3: XML results of hotel search

```

1. <%@ page language="java"%>
2. <%@ page errorPage="error.jsp" %>
3. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/mitl-1.0" prefix="mitl"%>
4. <% String title="Kayak mobile" %>
5. <mitl:doc title=<%=title%> stylesheet="kayak.css">
6.   <mitl:deck title=<%=title%> cardID="card1">
7.     <% String [] toFrmt = {"png","jpg","gif"};%>
8.     <mitl:image src="Kayak.jpg" alt="Kayak image" name="imgKayak"
          toFormat="<%= toFrmt %>"/> <mitl:break/>
9.     <mitl:bold>Welcome to Kayak mobile</mitl:bold>
10.    <mitl:form name="frmKayak" href="rsltKayak.jsp" method="post"
          midpElID="frmKayak" toMIDPEl="rsltKayak">
11.      Location: <mitl:field type="text" name="location" value="Prague"
          midpElID="txtLoc"/>
12.      Stars: <mitl:field type="text" name="stars" value="4" midpElID="txtStars"/>
13.      <mitl:field type="submit" name="btnSub" value="Submit"
          midpELID="btnSubmit"/>
14.    </mitl:form>
15.  </mitl:deck>
16. </mitl:doc>

```

Listing 9.4: Code of the start page of mobile Kayak service (kayak.jsp)

```

1. <%@ page language="java"%>
2. <%@ page errorPage="error.jsp" %>
3. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/mitl-1.0" prefix="mitl"%>
4. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/wstl-1.0" prefix="wstl"%>
5. <%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
6. <%@ taglib uri="http://java.sun.com/jstl/xml" prefix="x" %>
7. <% String title="Hotels" %>
8. <mitl:doc title=<%=title%> stylesheet="kayak.css">
9. <mitl:deck title=<%=title%> cardID="card2">
10. <c:set var="location" value="{param.location}"/>
11. <c:set var="stars" value="{param.stars}"/>
12. <wstl:rest URL="http://www.kayak.com/k/ident/apisession?
    token="fyUdBUY5Q0g5TXXYgKRcJg" returnType="parsed"
    parserClass="kayakSession" varName="SID" />
13. <wstl:rest URL="http://www.kayak.com/s/apisearch?basicmode
    =true&othercity={location}&apimode=1&_sid_={SID}"
    returnType="parsed" parserClass="kayakHotelsSID" varName="searchID"/>
14. <wstl:rest URL="http://www.kayak.com/s/apibasic/hotel?searchid=
    {searched}&c=20&m={stars}&_sid_={SID}
    returnType="XML" toFile="kayakHotels.xml"/>
15. <mitl:pagination>
16. <mitl:items>
17. <c:import var="xml" url="kayakHotels.xml" />
18. <c:import var="xslt" url="kayakHotels.xsl" />
19. <x:transform xml="{xml}" xslt="{xslt}" />
20. <mitl:items>
21. </mitl:pagination>
22. <mitl:break/>
23. <mitl:hyperlink href="kayak.jsp" name="Kayak Home" cardID="card1"
    midpElID="frmKayak">Find hotels<mitl:hyperlink/>
24. </mitl:deck>
25. </mitl:doc>

```

Listing 9.5: Document displaying the results of Kayak's search query (rsItKayak.jsp)



Figure 9.13: Mobile Kayak service in WML (Sony Ericsson T610)



Figure 9.14: Mobile Kayak service in XHTML (Nokia 6600)

Enterprise Data Web Service

The Enterprise Data Web Service is a relatively simple Web Service that provides information about available suppliers and products that can be delivered by these suppliers. Complete code of classes underlying the Web Service is included in listings 9.6-9.8. Listing 9.6 contains the code of the `Product` class. The `Product` object possesses various attributes such as `productID`, `description`, `name`, `price`, etc. These attributes are private and can be set and retrieved with appropriate setter and getter methods. It also has two constructors: one empty constructor and one with four parameters for product name, description, ID, and price. The `Supplier` class is similar to the `Product` class and consists of a set of private attributes, two constructors, and some setter and getter methods. The `SupplierInfo` class provides two methods: `getProductsBySupplier(String supplierID)` and `getAvailableSuppliers()`. The first method returns an array of `Product` objects, the second one of `Supplier` objects. Usually, information about products and suppliers is retrieved from a database. In the presented methods database interactions were replaced by a set of fixed suppliers and products.

The Enterprise Data Web Service was generated automatically from the described Java classes with the help of JDeveloper 10. The resulting WSDL file is shown in listing 9.9. The style of SOAP messages in this Web Service is `rpc` (as identified in line 45 in listing 9.9) and the endpoint address is `http://bj:8888/OrderList/SupplierInfoWS`. Listing 9.10 displays a sample SOAP request and listing 9.11 shows a corresponding SOAP response.

In order to provide the Enterprise Data Web Service on mobile devices, the MoWeSA framework is used. Listings 9.12 and 9.13 present the code of the start page for the service and a page with the results of the invocation of the Web Service method `getAvailableSuppliers()`. The starting page is generated with the help of MITL tags and consists of an image, some text and three links to pages that are based on the invocation of Web Services functions (cf. listing 9.12). The page displaying the data about suppliers is provided in a device-independent format using MITL and WSTL.

The main information about the Enterprise Data Web Service is provided in the `WSIdentificationTag` (cf. line 12 in listing 9.13) that creates the stubs for this Web Service. In the `WSOperationTag` the name of the method to be called is specified. The method does not contain any parameters; therefore the `WSInputTag` is not used. The method is then invoked using the

WSInvokeTag and the results of this invocation are stored in the `suppliers` variable. Since the returned results are in the form of an array, the elements constituting this array are extracted and used as title and description chunks for the `FragmentationTag` (cf. lines 17-23 in listing 9.13).

Figures 9.15 and 9.16 display the functionality of the Enterprise Data Web Service and the results of the invocation of the `getAvailableSuppliers()` method in WML/XHTML and as Java ME application, respectively.

```
1. package EnterpriseData;
2. import java.io.Serializable;

3. public class Product implements Serializable {
4.     private String productID = null;
5.     private String name = null;
6.     private String description = null;
7.     private double price = 0.0;

8.     public Product(){}
9.     public Product (String productID, String name, String description, double price){
10.         this.setProductID(productID);
11.         this.setName(name);
12.         this.setDescription(description);
13.         this.setPrice(price);
14.     }
15.     public String getProductID() {
16.         return productID;
17.     }
18.     public void setProductID(String productID) {
19.         this.productID = productID;
20.     }
21.     public String getName() {
22.         return name;
23.     }
24.     public void setName(String name) {
25.         this.name = name;
26.     }
27.     public String getDescription() {
28.         return description;
29.     }
30.     public void setDescription(String description) {
31.         this.description = description;
32.     }
33.     public void setPrice(double price){
34.         this.price = price;
35.     }
36.     public double getPrice(){
37.         return price;
38.     }
39. }
```

Listing 9.6: Product class (Product.java)

```

1. package EnterpriseData;
2. import java.io.Serializable;

3. public class Supplier implements Serializable {
4.     private String supplierID = null;
5.     private String supplierName = null;
6.     private String supplierDescription = null;

7.     public Supplier(){}
8.     public Supplier(String suppID, String suppName, String suppDesc){
9.         this.setSupplierID(suppID);
10.        this.setSupplierName(suppName);
11.        this.setSupplierDescription(suppDesc);}

12. public String getSupplierID() {
13.     return supplierID;
14. }
15. public void setSupplierID(String supplierID) {
16.     this.supplierID = supplierID;
17. }
18. public String getSupplierName() {
19.     return supplierName;
20. }
21. public void setSupplierName(String supplierName) {
22.     this.supplierName = supplierName;
23. }
24. public String getSupplierDescription() {
25.     return supplierDescription;
26. }
27. public void setSupplierDescription(String supplierDescription) {
28.     this.supplierDescription = supplierDescription;
29. }
30.}

```

Listing 9.7: Supplier class (Supplier.java)

```

1. package EnterpriseData;

2. public class SupplierInfo
3. {
4.     public Product [] getProductsBySupplier (String supplierID){
5.         Product [] prodBySupplier=null;
6.         if (supplierID.equals("1")){
7.             prodBySupplier = new Product[2];
8.             prodBySupplier [0]= new Product ("AB01", "IBM ThinkPad T30","", 433.0);
9.             prodBySupplier [1]= new Product ("AB02", "IBM ThinkPad R32","", 520.0);}
10.        if (supplierID.equals("2")){[...]
11.            [...]
12.        return prodBySupplier;
13.    }
14.    public Supplier [] getAvailableSuppliers(){
15.        Supplier [] availableSuppliers = new Supplier[3];
16.        availableSuppliers[0] = new Supplier("1", "IBM", "");
17.        availableSuppliers[1]= new Supplier("2","Microsoft","");
18.        availableSuppliers[2]= new Supplier("3","HP","");
19.        return availableSuppliers;
20.    }}

```

Listing 9.8: SupplierInfo class (SupplierInfo.java)

```

1. <?xml version = '1.0' encoding = 'UTF-8'?>
2. <definitions name="SupplierInfoWS"
   targetNamespace=http://Orders/SupplierInfo.wsdl
   xmlns=http://schemas.xmlsoap.org/wsdl/
3.   [further namespaces definitions for xsd, soap, tns and ns1] >
4.   <types>
5.     <schema targetNamespace="http://Orders/SupplierInfoWS.xsd"
6.       xmlns="http://www.w3.org/2001/XMLSchema"
7.       xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
8.     <complexType name="ArrayOfOrders_Product"
9.       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
10.      <complexContent>
11.        <restriction base="SOAP-ENC:Array">
12.          <attribute ref="SOAP-ENC:arrayType"
13.            wsdl:arrayType="ns1:Orders_Product[]" />
14.        </restriction>
15.      </complexContent>
16.    </complexType>
17.    <complexType name="Orders_Product" jdev:packageName="EnterpriseData"
18.      xmlns:jdev="http://xmlns.oracle.com/jdeveloper/webservices">
19.      <all>
20.        <element name="productID" type="string"/>
21.        <element name="name" type="string"/>
22.        <element name="description" type="string"/>
23.        <element name="price" type="double"/>
24.      </all>
25.    </complexType>
26.    <complexType name="ArrayOfOrders_Supplier"
27.      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
28.      [...]
29.    </complexType>
30.    <complexType name="Orders_Supplier" jdev:packageName="EnterpriseData"
31.      xmlns:jdev="http://xmlns.oracle.com/jdeveloper/webservices">
32.      [...]
33.    </complexType>
34.  </schema>
35. </types>
36. <message name="getProductsBySupplierRequest">
37.   <part name="supplierID" type="xsd:string"/>
38. </message>
39. <message name="getProductsBySupplierResponse">
40.   <part name="return" type="ns1:ArrayOfOrders_Product"/>
41. </message>
42. <message name="getAvailableSuppliersRequest"/>
43. <message name="getAvailableSuppliersResponse">
44.   <part name="return" type="ns1:ArrayOfOrders_Supplier"/>
45. </message>
46. <portType name="SupplierInfoPortType">
47.   [...]
48. </portType>
49. <binding name="SupplierInfoBinding" type="tns:SupplierInfoPortType">
50.   <soap:binding style="rpc"
51.     transport="http://schemas.xmlsoap.org/soap/http"/>
52.   [...]
53. </binding>
54. <service name="SupplierInfoWS">
55.   <port name="SupplierInfoPort" binding="tns:SupplierInfoBinding">
56.     <soap:address location="http://bj:8888/OrderList/SupplierInfoWS"/>
57.   </port>
58. </service>
59. </definitions>

```

Listing 9.9: WSDL file

```

1. <?xml version = '1.0' encoding = 'UTF-8'?>
2. <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3.   <SOAP-ENV:Body>
4.     <ns1:getAvailableSuppliers xmlns:ns1="SupplierInfoWS"
       SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
5.   </SOAP-ENV:Body>
6. </SOAP-ENV:Envelope>

```

Listing 9.10: SOAP request in Enterprise Data Web Service

```

1. <?xml version = '1.0' encoding = 'UTF-8'?>
2. <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3.   <SOAP-ENV:Body>
4.     <ns1:getAvailableSuppliersResponse xmlns:ns1="SupplierInfoWS"
       SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
5.     <return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
       xsi:type="ns2:Array" xmlns:ns3="http://Orders/SupplierInfoWS.xsd"
       ns2:arrayType="ns3:Orders_Supplier[3]">
6.       <item xsi:type="ns3:Orders_Supplier">
7.         <supplierDescription xsi:type="xsd:string"/>
8.         <supplierID xsi:type="xsd:string">1</supplierID>
9.         <supplierName xsi:type="xsd:string">IBM</supplierName>
10.      </item>
11.      <item xsi:type="ns3:Orders_Supplier">
12.        <supplierDescription xsi:type="xsd:string"/>
13.        <supplierID xsi:type="xsd:string">2</supplierID>
14.        <supplierName xsi:type="xsd:string">Microsoft</supplierName>
15.      </item>
16.      <item xsi:type="ns3:Orders_Supplier">
17.        <supplierDescription xsi:type="xsd:string"/>
18.        <supplierID xsi:type="xsd:string">3</supplierID>
19.        <supplierName xsi:type="xsd:string">HP</supplierName>
20.      </item>
21.    </return>
22.  </ns1:getAvailableSuppliersResponse>
23. </SOAP-ENV:Body>
24. </SOAP-ENV:Envelope>

```

Listing 9.11: SOAP response in Enterprise Data Web Service

```

1. <%@ page language="java"%>
2. <%@ page errorPage="error.jsp" %>
3. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/mitl-1.0" prefix="mitl"%>
4. <% String title="EDS" %>
5. <mitl:doc title=<%=title %>>
6.   <mitl:deck title=<%=title%> cardID="card1">
7.     <% String [] toFrmt = {"png","jpg","gif"};%>
8.     <mitl:img src="EDS.jpg" alt="EDS image" name="imgEDS" toFormat=<%=toFrmt%>/>
9.     <mitl:bold>Welcome to Enterprise Data Services</mitl:bold><mitl:break/>
10.    <mitl:hyperlink href="suppliers.jsp" name="Suppliers" cardID="card1"
        midpElID="frmSuppliers">Find suppliers</mitl:hyperlink>
11.    <mitl:break/>
12.    <mitl:hyperlink href="products.jsp" name="Products" cardID="card2"
        midpElID="frmProd">Find products</mitl:hyperlink>
13.    <mitl:break/>
14.    <mitl:hyperlink href="orders.jsp" name="Orders" cardID="card3"
        midpElID="frmOrders">Find orders</mitl:hyperlink>
15.    <mitl:break/>
16.  </mitl:deck>
17. </mitl:doc>

```

Listing 9.12: Welcome page for Enterprise Data Web Service

```

1. <%@ page language="java"%>
2. <%@ page errorPage="error.jsp" %>
3. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/mitl-1.0" prefix="mitl"%>
4. <%@ taglib uri="http://www.uni-ffo.de/~euv-6204/taglibs/wstl-1.0" prefix="wstl"%>
5. <%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
6. <% String title="Suppliers" %>
7. <mitl:doc title=<%=title%>>
8.   <mitl:deck title=<%=title%> cardID="card1">
9.     <% String [] toFrmt = {"png","jpg","gif"};%>
10.    <mitl:img src="EDS.jpg" alt="EDS image" name="imgEDS"
        toFormat=<%=toFrmt%>/>
11.    <mitl:bold>Suppliers found in the database:</mitl:bold><mitl:break/>
12.    <wstl:wsit wsID="EDS" type="rpc"
        wsdlLocation="http://Orders/SupplierInfo.wsdl"
        endpoint="http://bj:8888/OrderList/SupplierInfoWS"
        binding="SupplierInfoBinding">
13.      <wstl:operation operationID="EDS_Suppliers"
        operationName="getAvailableSuppliers"/>
14.      <wstl:invoke resultID="suppliers" wsID="EDS" operationID="EDS_Suppliers"/>
15.    </wstl:wsit>
16.    <mitl:FragmntOn>
17.      <c:forEach var="supplierName" items="${suppliers.supplierName}">
18.        <mitl:title>${supplierName}</mitl:title>
19.      </c:forEach>
20.      <c:forEach var="edsSuppliers" items="${suppliers.supplierDescription}">
21.        <mitl:desc>${edsSuppliers}</mitl:desc>
22.      </c:forEach>
23.    </mitl:FragmntOn>
24.  </mitl:deck>
25. </mitl:doc>

```

Listing 9.13: Page with results of EDWS



Figure 9.15: Enterprise Data Web Service in WML (Nokia 6610) and XHTML (Siemens SX1)

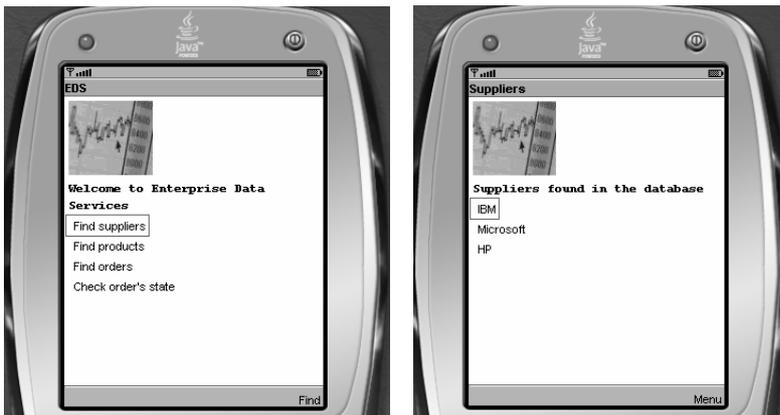


Figure 9.16: Enterprise Data Web Service as Java ME application

9.8 Evaluation of the approach

The same groups of users that participated in the questionnaire about MITL and its IDE were also asked to assess the quality of the MoWeSA framework and the enhanced IDE. Similarly to the previous evaluation, both groups (students and professional programmers) had to develop a simple and a complex mobile application based on Web Services. The results of the questionnaire are summarized in table 9.5, the questionnaire itself is included in Appendix 6.

In the first part of the survey, users had to evaluate the Web Services Tag Library. Both groups of users considered WSTL as sufficiently powerful for the development of simple and complex mobile applications. WSTL was a preferred solution in comparison with the development of mobile applications based on Web Services with the help of programming languages such as Java or VB.Net and their respective libraries offering support for Web Services. The surveyed students had serious doubts whether non-experienced programmers would be able to use WSTL even though its syntax resembles other commonly known markup languages. Professional developers were not so skeptical and found WSTL relatively easy to apply even by occasional users. Generally, it was evident that the students encountered serious problems in the development of mobile Web Services, mainly due to the lack of knowledge about the standards underlying Web Services. They succeeded in the communication with REST Web Services because it was relatively easy for them to understand that the operation and its parameters are passed to a Web Service in the form of an URI and the response is returned in XML.

In the second part of the questionnaire, the groups had to evaluate the usability and quality of the MoWeSA framework and the enhanced IDE. The Integrated Development Environment for MoWeSA collected positive judgments. Both groups of users liked the product and were of the opinion that it makes the development of mobile Web Services much easier and eliminates the need for professional knowledge of the WSTL syntax. They found the attached Web APIs and the context-sensitive help for completing WSTL tags for chosen Web Services a particularly good function.

As far as the MoWeSA framework is concerned, inexperienced programmers had serious difficulties with the generation of complicated applications that required more advanced knowledge of the underlying standards and programming libraries e.g. of JSTL or Web Services. They evaluated therefore MoWeSA as a framework that is quite complex and difficult to learn. They failed to develop device-independent applications that would integrate many sources of information, were not able to parse XML with the help of JSTL, could not cope with arrays returned as results of Web Services calls, etc. Their general conclusion was that the MoWeSA framework is not suitable for end-users without any previous programming experience and that the complexity of the framework discourages them from using it. Although professional developers were able to develop mobile applications with the help of MoWeSA they also claimed that a lot of specific knowledge is necessary to succeed in the development.

Question	Average Students	Standard Deviation Students	Average Developers	Standard Deviation Developers
A1	6.2	0.4	6.0	0.48
A2	5.98	0.52	5.5	0.71
A3	6.4	0.62	6.1	0.66
A4	2.5	0.8	1.5	0.62
A5	5.4	0.4	6.6	0.31
A6	5.0	0.91	6.1	0.34
A7	4.34	0.92	6.0	0.57
A8	5.5	0.52	5.9	0.89
A9	5.58	0.76	6.2	0.63
A10	2.0	0.54	5.5	0.73
B1	3.6	0.67	6.0	0.49
B2	5.5	0.89	6.2	0.34
B3	5.1	0.45	5.9	0.64
B4	2.0	0.65	1.6	0.23
B5	4.5	0.94	5.8	0.34
B6	3.0	0.55	2.2	0.41
B7	4.5	0.35	5.2	0.62
B8	3.0	0.92	5.5	0.45
B9	5.5	0.30	6.1	0.34

Scale:
1. Strongly Disagree 2. Moderately Disagree 3. Mildly Disagree 4. Neither Disagree nor Agree 5. Mildly Agree 6. Moderately Agree 7. Strongly Agree

Table 9.5: Evaluation of the MoWeSA framework and its IDE (averages and standard deviations for two groups of users, cf. Appendix 6 for questions)

10

Conclusions and future work

This dissertation is concluded with a short summary of the work, an overview of the main theses, and further research possibilities.

10.1 Summary and key theses of the dissertation

Mobile computing with its permanent access to information offers new opportunities for private and business users. At the same time, it also poses several challenges that do not exist in personal computing based on the desktop metaphor. Although progress has been made in “porting” traditional desktop and Web-based applications to mobile devices, the screen size factor and the limited input/output possibilities of wireless devices force developers and researchers to radically rethink user interfaces and the methods of user interactions. A parallel mobile universe, consisting of wireless versions of existing Web pages or desktop applications, is neither economically efficient nor technically feasible. Device-specific content delivery cannot be achieved by a miniaturization of existing graphical desktop interfaces or Web pages. This method completely ignores the significance of mobile context.

Developers and researchers worldwide work on new solutions that would enable device-independent content delivery to different devices. The research focus is put on matching the content with capabilities and limitations of the delivery environment. A further goal is to maximize the number of devices to which the content can be delivered. Various adaptation approaches were developed so far but most of them share similar drawbacks. Usually they are too sophisticated for an end-user programmer, require a good knowledge of a particular framework, are either prototypical or proprietary, or cannot be applied to existing content. Therefore, the main challenge in this research work was to develop new frameworks for automated content adaptation to mobile devices. These frameworks are based on open-source technologies, facilitate the development of mobile applications by dividing tasks between developers and content designers, and do not require extensive programming experience (unless somebody wants to extend them).

The first approach - the framework based on Mobile Interfaces Tag Library - can be applied to various forms of content. Text and graphics can be taken from existing Web pages, content syndication formats, databases or entered manually. It generates output in HTML, XHTML, WML, and Java ME. The available syntax can be extended by adding further tags to the library.

The Mobile Web Services Adaptation (MoWeSA) framework focuses on delivering Web Services to mobile thin and smart clients. It supports RPC-style and document-style Web Services as well as the Representational State Transfer (REST) Web Services. The WSTL and MITL libraries are supported

by an Integrated Development Environment (IDE) and were positively evaluated by inexperienced users and programmers with a profound knowledge of Java EE.

Eighteen main theses were formulated as answers to the objectives of this research work introduced in section 1.2:

- 1) The developers of mobile services are confronted with various formats for mobile devices. There is no common standard for the development of wireless solutions, although most advanced phones are provided with a support for XHTML MP/WCSS and Java ME. The adoption of one of these formats as a universal technology is difficult since they are not device-independent and are rendered differently by various browsers/phones. Similarly, the supported multimedia formats (audio, video, graphics) vary depending on the device (cf. chapter 2.2).
- 2) With the growing popularity of 3G networks with high transmission rates and increasing capabilities of mobile devices, mobile services have better chances of achieving commercial success. With a plethora of mobile devices available on the market, it is crucial for the adaptation process to retrieve the information about the most important features of devices. Unfortunately, handsets send incorrect data about their capabilities and sometimes present content in a way nobody would expect. They claim, for example, to be Netscape 4.0 compatible, have odd page-size limitations, and formatting quirks. It is still not possible to obtain all the data needed with the help of databases of devices such as WURFL or by using information stored in CC/PP profiles. A complete list of existing devices is not available. Most device profiles are Euro-centric, missing the information about phones produced in other parts of the world. A feasible solution to these problems gives the so-called "Pareto's principle" (the 80-20 rule)¹⁴¹. Applied to the mobile devices, the rule states that 20 percent of the most popular browsers/phones account for 80 percent of the mobile Web access. Therefore, if developers focus on the features of the most popular phones, they will cover the majority of the market (cf. chapter 7.1).
- 3) Contrary to the expectations of analysts, the mobile Internet has not repeated the incredible success of the Internet. Although the penetration rates for mobile phones are very high, the acceptance of the mobile Internet is still low. The main adoption barriers identified by private users are poor mobile content, inconvenient user interfaces (difficult navigation and browsing), insufficient transfer speed and its quality, security aspects, high grade of mobile browsing complexity, and exorbitant costs of mobile browsing (in comparison with the traditional Internet browsing). Business users are reluctant to adopt mobile solutions due to open security questions, insufficient quality of services, lack of mature technology standards, expensive integration with existing infrastructure, lack of specific knowledge in the area of mobile solutions, and high costs (cf. chapter 3.3).

¹⁴¹ In 1906, Italian economist Vilfredo Pareto observed that twenty percent of the people owned eighty percent of the wealth. The name "Pareto's principle" was inaccurately attributed to the universal 80/20 rule by Joseph M. Juran. In his publication, he called it also a principle of „vital few and trivial many". The general observation was that only 20 percent of something is always responsible for 80 percent of the results [cf. Jura60; Jura75].

- 4) Currently available mobile content is scarce compared with 11.5 billion of traditional Web pages. The quality of the content is not sufficient to attract new users. The most common mistakes of mobile content authors are: inadequate formatting (e.g. page elements distributed all over the screen, large font sizes), the use of very large images that made download very slow or images in incorrect formats, placing too much content on one screen and forcing users to scroll horizontally, referring to dead links or links that could not be displayed in mobile browsers (cf. chapters 3.3 and 5.1).
- 5) The traditional Internet is the most dangerous rival of mobile browsing. It is perceived by users as a cheaper and more convenient substitute for the wireless Internet. Mobile Internet services have a chance to succeed if they are able to offer some added value in comparison to Internet-based services. Alternatively, they can compete with Web services if their main target group are mobile users and it is impossible to use the Internet in such situations. Examples of such applications are localization services (e.g. instance routing) or “roadside“ services such as vending/parking machine payments (cf. chapter 3.3) .
- 6) Mobile Web usage is highly service- and context-driven. Therefore, merely the elimination of technical obstacles (e.g. by providing better adaptation approaches) does not guarantee higher adoption rates of mobile Web services. The best proof for this thesis is Japan, with one of the highest rates of mobile services usage. Although mobile applications in Japan have to be written in various languages because each provider sticks to its own standard, mobile browsing is very popular. The key success factor is the development of context-sensitive and user-oriented applications and appropriate marketing and business models (cf. chapters 3.3, 4.4 and 5.1).
- 7) The failure of mobile Internet outside of Japan and Korea can be contributed to the lack of cooperation between service providers, content and application providers, and handset manufacturers. Service providers do not want to share their revenues with content providers and prefer to earn money from SMS and entertainment services than from push-based Internet mail or content accessed via the input of URLs. Government regulations seem to be the only way to force service providers to adopt a more open approach towards mobile Internet (cf. chapter 3.3)
- 8) Mobile content providers and business users are aware of the potential of mobile computing. However, all of them lack a concept how to use this potential. Managers admit that mobile applications can significantly improve corporate business agility. They can even name the main benefits for their companies. Most of them, however, are not able to formulate any clear business strategy for the introduction of mobile solutions. The mobile industry, on the other hand, is looking for killer applications that would rapidly be adopted on a mass scale and would therefore generate high revenue. This approach seems to be wrong because of its strong focus on technical aspects. Mobile services providers should rather look for killer user experiences (cf. chapter 3.3).
- 9) Device-independent solutions can significantly improve the development process of mobile solutions and can diminish their cost and complexity. It is expected that mobile services will increase their market share in the future and will therefore be able to generate more revenues.

These expectations are confirmed by recent studies that report on the increased usage of mobile multimedia content in 2005. Mobile users are more willingly browsing the mobile Internet (most of them have already visited their mobile operator's portal). This is due to the availability of more sophisticated devices supporting HTML/XHTML and Java ME and improved network quality. The business usage of mobile applications also awakes hopes for the revenue boost. Since it is not reasonable to follow the multiple authoring approach for further mobile services, single-source authoring will considerably gain importance with the growing usage of mobile content (cf. chapter 3.3)

- 10) The adaptation to the features of various mobile devices is a complex process and requires consideration of many different aspects. The main subjects of the adaptation process are style, layout, structure, navigation, application interactions, content, and transfer of data. For all these elements specific adaptation techniques were developed (cf. chapter 4.1). Furthermore, the design of mobile user interfaces requires the consideration of design principles that are unique for the mobile environment. The W3C Mobile Web Best Practices recommendation summarizes the most important aspects of mobile design and development. Content that is relevant and easy to navigate as well as the medium's ability to personalize content are the most important factors for creating wireless interfaces that will be adopted by a wide audience (cf. chapter 4.4).
- 11) Adaptation approaches should particularly fulfill the Device Independence Principles and Authoring Challenges as specified by the W3C. The Device Independence Principles focus on three perspectives: user's, author's and delivery points of view. They take into account device-independent access, device independent Web page identifiers, functionality, incompatible access mechanisms, and harmonization. The Authoring Challenges were developed for different types of authors participating in the adaptation process. They refer to different requirements for device-independent approaches such as comprehensiveness of scope, smooth extensibility of applications, simplicity, abstraction, variability of delivery context, author-specified variability (with regard to content, media, navigation, and layout), client-side processing, use of existing standards, context-awareness of applications, and affordability (cf. chapter 4.1).
- 12) Adaptation techniques can be evaluated with regard to the commonly known ISO 9126 quality standard that makes a distinction between external/internal quality and quality in use. The quality in use helps to evaluate user's perception and acceptance of the software. It includes four features: effectiveness, productivity, safety, and satisfaction. The quality in use can be comprehensively assessed with the help of the QUIM model that includes ten usability factors (e.g. efficiency, learnability, universality, usefulness, accessibility, etc.) and splits them further into twenty six sub-factors (cf. chapter 4.3).
- 13) Four types of adaptation methods can be distinguished: manual authoring, scaling, transcoding and transforming. Transforming and transcoding can deliver the best results but are usually quite complex. Both methods show certain trade-offs between their complexity/comprehensibility and affordability. These factors are highly correlated: the more comprehensive the approach, the higher development and maintenance costs it generates. Furthermore, adaptation approaches can be classified according to the controller of the adaptation process as intermediate, client-

side and server-side adaptation. Server-side adaptation approaches are enjoying growing popularity among researchers and practitioners. They can be further divided into approaches based on XML/XSLT, User Interface Description Languages and frameworks based on the Model-View-Controller design pattern (cf. chapters 5 and 6).

- 14) With regard to the Device Independence Authoring Challenges, server-side adaptation approaches gained the highest scores. Most of them offer a very good support for the variability of delivery context and author-specific variability. They support the generation of different navigation mechanisms, enable the presentation of various media types and content depending on the device type. In these approaches, the layout can vary and presentation units can be aggregated or defragmented (split into smaller units), depending on the delivery context. The frameworks are extensible and it is possible to scale the complexity of applications. However, the application scope is relatively small but the complexity of such frameworks and the effort required to learn them are very high. Although server-side adaptation frameworks were evaluated as the best techniques, client-side adaptation approaches with a limited support for the variability of delivery context and author-specific variability are more willingly used. This fact provides evidence for a well-known phenomenon that simple, scalable, and easily affordable approaches are more successful and popular than complicated techniques with numerous functionalities (cf. chapter 6.4).
- 15) Adaptation approaches help to reduce development and maintenance costs. They usually offer some kind of abstraction from underlying devices, an instant support of new devices and better usability for developers and content creators. Existing adaptation approaches, however, have different drawbacks. Client-side adaptation techniques are computationally intensive, can be applied only for specific markups and are supported by a limited set of devices. Intermediate approaches lack knowledge about the structure of converted sites and often provide incorrect results. They are also applicable to a limited number of languages and cannot sometimes be extended. Server-side adaptation approaches are the most complex and complicated ones. Developers try to avoid these frameworks because they are very sophisticated and significant effort is needed to learn them (cf. chapter 5.2 and 6.4).
- 16) The developed approaches (the MITL and MoWeSA frameworks) belong to the server-side adaptation category and are relatively easy to learn and apply. They were designed for experienced and occasional end-user developers. The frameworks fulfill the requirements of high user-friendliness, learnability and usability because these three factors can guarantee a greater adoption of these approaches and can contribute to the increase in the number of mobile applications. An IDE supports the authors of wireless solutions in the development of device-independent applications based on MITL and MoWeSA. The frameworks use open-source technologies and can be easily extended with further formats. Clear separation of tasks between Web page designers and developers enables effortless maintenance of the software and offers further development possibilities. Simplicity and scalability of effort are the most important advantages of both frameworks (cf. chapters 8.7 and 9.8).

- 17) Mobile Interfaces Tag Library is an approach that is based on the concept of JSP Tag Libraries and possesses the structure of XML. Tags and attributes in MITL resemble markup languages such as WML, XHTML, or HTML. This helps developers to apply this language intuitively even if they are not familiar with its syntax. MITL delivers output in (X)HTML, WML, and Java ME, depending on the format supported by a mobile device. It can also produce HTML for standard desktop browsers. The generation of Java ME applications is, however, particularly interesting because there exist currently only one prototypical approach that is able to output this format. All other frameworks focus on standard markups such as WML or XHTML (cf. chapter 6.3). Since MITL documents are actually JSP pages with the syntax of MITL, it is possible to combine the tags from MITL with tags from other available libraries. This increases the number of potential mobile applications because many Web services encapsulate their basic functionalities in the form of tag libraries. Additionally, the MITL tags can also be mixed with pure Java code (cf. chapter 8.4).
- 18) The Mobile Web Services Adaptation (MoWeSA) framework supports three types of Web Services. The greatest importance is currently attributed to the REST-based Web Services, mainly due to the simplicity of request generation. The MoWeSA approach applies MITL for the generation of end-formats and the Web Services Tag Library (WSTL) takes care of the communication with Web Services. This framework is also based on the concept of JSP Tag Libraries. Interestingly, Sun is going to develop a JSP Tag Library for Web Services in the future, but does not take into consideration providing a tag library that would enable a communication between mobile devices and Web Services (cf. chapter 9).

Even though the proposed frameworks eliminate some disadvantages of other adaptation approaches, it should be emphasized that the wide adoption of mobile applications is still dependent on various additional factors that are not of technical nature.

10.2 Further research and development possibilities

This section outlines further development possibilities for both adaptation frameworks. It also focuses on issues that may influence the design and development of mobile solutions and adaptation approaches in the future.

Possible improvements for the MITL and MoWeSA frameworks

Most mobile pages possess a relatively simple structure. For example, content authors do not use tables as layout elements and do not squeeze many different elements in one row like in traditional Web pages. They rather place them in a sequential manner and the user has to scroll down to see more content. Therefore creating a mobile page for various devices in different markup languages does not require complicated transcoding heuristics or automatic summarization techniques.

Problems arise if existing Web sites have to be converted to fit mobile screens. Their navigational structures, the amount of content per screen and the layout strongly diverge from mobile versions. It is much easier to assemble a Web site from mobile pages than to convert existing HTML pages to

wireless sites because of insufficient separation between logic, data and presentation. MITL is not suitable for automatic conversion of existing Web pages into their mobile versions because it lacks mechanisms for recognizing the structure of documents, generating summaries or extracting keywords from large fragments of text. It offers, however, the possibility to extract content manually from Web sites and to make use of it in mobile page. Alternatively, RSS feeds with selected information from a Web page can be applied for this purpose. As the popularity of syndication formats is growing, MITL should offer better support for them. New formats, such as Atom, should be included as input formats and the existing support for RSS should be extended. It is particularly important to add a tag's support for the RSS namespaces since end-users often have problems with XPath and XSLT.

Another drawback of the current version of the MITL framework results from the insufficient support of delivery context by device manufacturers. The framework does not use its own database with profiles of devices; instead of this it relies on the information about device characteristics delivered automatically by the mobile handsets. Since many producers do not offer such support but usually deliver data about the phone type, it would be reasonable to create own database with device profiles or to use additional collections of profiles such as Wireless Universal Resource (WURFL).

The MITL framework supports the most popular formats for mobile devices: HTML, XHTML with WCSS, WML, and Java ME. It is also able to convert one image type to another. The capability to change the image format is particularly important, since Java ME applications accept only PNG images and most devices offer a limited support for different image types, usually sticking to one or two formats. In its further versions, the MITL library could be extended with additional formats. Particularly interesting would be the addition of languages for vocal-based and multimodal interfaces (VoiceXML, SALT, XHTML+Voice) and support for multimedia formats (e.g. SMIL, SVG, etc.).

The Integrated Development Environment for MITL and MoWeSA could be rewritten as an Eclipse plug-in. Eclipse¹⁴² is an extensible platform that supports programming tasks and enables the development of Java-based solutions, including IDEs with the Eclipse's look and feel. New plug-ins, encapsulating innovative functionalities, can be added to Eclipse and removed from it easily. The integration mechanism gives developers the possibility to include additional components/tools into its programming environment and to benefit from the work of other programmers using one integrated platform. Plug-ins can also exist as stand-alone applications. Rich-client applications that are based on the open-source Eclipse Rich Client Platform (RCP) are quite easy to develop with the help of the Standard Widget Toolkit (SWT) [GaBuMc03, pp. 219-306]. SWT is a set of widgets and graphics libraries integrated with a native window system and offering an OS-independent API. It maps directly to the native graphics of an operating system. The Eclipse RCP provides a standardized component model supporting movable and stackable window components such as editors and views, menus, tool bars, tables, trees, etc. Additionally, a consistent and native look and feel for applications is offered. In the RCP the user interface is separated from business logic and object model, similarly to the MVC model.

¹⁴² <http://www.eclipse.org>.

The IDE could further be enhanced by adding a context-sensitive help to it. This functionality would enable the users to see hints on tags' attributes, their possible values, syntax errors, etc. An additional view displaying available database and server connections could also be included in the editor.

In the MITL library, the presentation could be completely separated from the content by storing text and the addresses of images in XML files and using appropriate references to them in MITL tags. A `LayoutTag` could be introduced to enable layout variety and better aggregation/decomposition of content. The `LayoutTag` would suggest different vertical and horizontal arrangements of content and the allocation of content chunks by references to XML fragments.

The generated Java ME applications could further be improved by using the J2ME Polish library¹⁴³ that provides a collection of components for advanced, "polished" mobile Java applications. The library makes it possible to develop alternatives to the high-level Mobile Information Device Profile (MIDP) user interfaces. The GUIs are designed outside the MIDlet code with the help of simple Cascading Style Sheets (CSS). This enables a better positioning of elements and introduces colors and formatting to boring standard Java ME interfaces.

Context-aware computing

Mobile applications should take into account different types of context such as computing context, user context, physical context, and time context instead of focusing exclusively on the delivery context. Context-sensitive or context-aware applications can discover and take advantage of contextual information. They were categorized by Schlitt as follows [cf. ShAdWa94]:

- proximate selection applications – in those applications the objects located nearby are easier to choose
- programs with automatic contextual reconfiguration – depending on context changes new components can be added, existing components removed, or the connections between components can be changed
- applications with contextual information and commands – they can produce various results depending on the context
- applications with context-triggered actions – are based on simple conditional statements that specify how a system should adapt to context changes.

It is crucial for mobile applications to possess the ability to adapt to users' information needs. The users should be able to benefit from context changes without paying attention to them and without undertaking additional efforts to make the applications context-sensitive (e.g. by providing additional information about their locations, time, work schedule, etc.). Mobile users are either people in motion (in a train, bus, etc.) or people working with mobile devices, usually in a non-office environment. This

¹⁴³ <http://www.j2mepolish.org/>.

implies that they have reduced cognitive abilities and are willing to work with applications that do not require high concentration and do not overwhelm the user with information [cf. Tee05].

Context-aware computing has already been proposed a decade ago but context-aware applications were mainly developed for research purposes [cf. ChKo00; Hueb+05]. They have never been applied on a mass scale. This is going to change as more and more enterprises are focusing on the development of context-sensitive applications. For example, NeoMedia Technologies offers a patented solution named “qode” that enables a match between specific products and services with appropriate mobile Web pages¹⁴⁴. The users can be automatically redirected to mobile pages related to a specific product after the mobile device captured a picture of its barcode or its advertisement in a newspaper. This technology opens new areas of applications: it is possible to buy a ticket for a movie from its poster, to find the lowest price for a book by scanning its barcode and to buy this book online, to find concert dates from a CD cover, etc. For the solutions developed in this research project, the growing popularity of mobile context-aware applications implies the necessity of further research in this area. This research would include the development of formats for describing contextual information and the integration of such information within the frameworks.

Semantic adaptation

The Semantic Web is based on the idea of universal representation of concepts and their interrelationships that do not encompass only one particular system, application or organization. It is possible to perform automatic logic reasoning, for example with the help of artificial intelligence (AI) technologies, basing on the formats of Semantic Web (e.g. OWL, RDF, etc.). With the growing popularity of the Semantic Web, semantic adaptation is also gaining more and more attention.

Semantic adaptation was already applied to a certain extent in the annotation-based transcoding and is regarded as a very promising adaptation technique. In semantic adaptation, original resources are adapted according to information that provides descriptions of these resources (the so-called metadata). The application of semantic adaptation approaches can be facilitated by the fact that many enterprises already possess some kind of data that describe the structure of the content and the relationships between information fragments. Such data is typically stored in an application data dictionary being part of a database. In the application data dictionary content chunks are associated with GUI elements such as fields, labels, etc. Application data dictionaries are, however, not sufficient to enable semantic-based adaptation. They mix content with presentation elements and do not provide mechanisms for navigational mapping.

Approaches used in adaptive hypermedia such as Semantic Hypermedia Design Method (SHDM) [MoSc04; Schw+04] are more appropriate for device-independent adaptation. SHDM separates conceptual and navigational design and consists of various artifacts that correspond to different design phases (as displayed in table 10.1). In this approach, five design steps can be distinguished:

¹⁴⁴ <http://www.paperclick.com/applications.jsp>.

requirements gathering, conceptual design, navigational design, abstract interface design and implementation. Conceptual design focuses on objects and behaviors in the application domain. Navigational design takes into account users' profiles and tasks. Navigation objects are views on conceptual objects. In the navigation design phase, it has to be specified which nodes can be reached by users and which relationships exist between these navigation objects, what is the set of objects used in a particular context and how to access this set, and finally, what content should be displayed depending on the user's context. The abstract interface model supports the exchange of information between the user and the application and activates functionalities. Any interface is a combination of predefined abstract interface widgets. Abstract interface widgets can react to external events, display some content, receive a variable's value or can be composed of these widget types. Abstract interface widgets are mapped to concrete widgets; the mapping is defined in the abstract interface ontology. This ontology provides also the mapping between abstract widgets and navigation elements.

The implementation is based on JSP pages and tag libraries. The JSP pages represent a mixture of abstract interface widgets, navigation elements, and data. The code behind tags implements the mapping between abstract interface widgets and concrete interface widgets. It consults the mapping ontology and generates appropriate interface code. Additional parameters such as device properties may also be taken into account and can influence the output.

Artifact	Definition Language	Description
Conceptual ontology	OWL-DL with annotations and additional SHDM rules	Conceptual class definition
Conceptual instances	Conceptual ontology	Application data defined according to the conceptual ontology
Navigational mapping	Navigational mapping definition vocabulary	Rules mapping conceptual classes into navigational classes
Navigation space definition	Navigation space definition vocabulary	Definition of the navigational elements – contexts and access structures (indexes)
Navigational ontology	OWL-DL	Navigational class (node) definition
Navigational instances	Navigational ontology	Application data defined according to the navigational ontology
Abstract interface	Abstract interface definition vocabulary	Abstract interface definition, including abstract interface elements and their mapping to the navigation model and to concrete interface widgets
Concrete interface widget ontology	Definition vocabulary for concrete interface widgets	Definition of possible concrete interface widgets to be used in the implementation

Table 10.1: SHDM artifacts [Schw+04]

Similar approaches based on the introduction of an additional semantic layer can be encountered in many adaptive hypermedia systems [cf. Andr05; PaRePa04]. In such systems two categories of adaptation methods can be found: content level adaptation and link level adaptation methods [BaJe06; Brus03]. Both method types apply semantics for the description of content meaning, relationships between content fragments or for providing additional information about the navigational structure. For

example, in the Morfeo's Semantic Mobility Channel (MorfeoSMC) content is described semantically and is subsequently bound by references to visual controls (GUI elements) [Cant+06].

The main problem in semantically-oriented approaches is the lack of common terminology. Different system designers use diverse metadata for description of identical content. Universally accepted ontologies are still missing and no common format for expressing ontologies exists.

Semantic adaptation is certainly a future-oriented approach and enables advanced adaptation of mobile applications. It strictly separates content from interface elements and navigational structure, enabling the presentation of the same content in different ways, using various GUI elements and navigation structures depending on the capabilities of devices. However, its complexity and possible implementation problems are still severe obstacles for rapid adoption of such techniques.

Mobile Web 2.0 and AJAX

The term Web 2.0, coined by O'Reilly Media in 2004, refers to a novel generation of Web services that allow users to collaborate in new ways, share existing information and re-use the available services in other applications. The concept encompasses many aspects and is sometimes considered a new marketing buzzword that simply summarizes the technologies that were available before. The most relevant concepts of Web 2.0 were introduced by Tim O'Reilly [cf. ORei05]. O'Reilly pointed out the significance of unique and hard-to-recreate sources of data (data as the next "Intel inside") and the importance of users' contribution to the development of Web 2.0 (users add value to services). In Web 2.0, applications are considered as services that are constantly modified and tested and not as finished software products. They remain at the stage of "the perpetual beta" through their entire lifecycle. The services are loosely-coupled and reusable. Web interfaces and content syndication mechanisms make it possible to use the existing content in other applications. New services created from existing applications embody the "innovation in assembly" concept. According to this idea, it is even possible to create value out of abundant applications if they are assembled in new and effective ways. Furthermore, the offered services should be available from different devices and not only from desktop browsers (the concept of "software above the level of a single device").

Web 2.0 aims at harnessing the so-called "long tail" and collective intelligence. The "long tail" is a metaphorical expression and refers to the immeasurable number of small Web sites (the "tail") contrasting with a few sites that are of great importance (the "head"). The collective intelligence includes concepts and terms like peer production, the wisdom of crowds and the network effect. Peer production [Benk02] is a new model of production, in which the central role is attributed to people who participate or create projects (mainly Internet-based) without financial rewards or organizational structure. The concept of the wisdom of crowds [cf. Suro04] assumes that large groups of people are more intelligent than the elite of few, brilliant persons. Groups are able to solve problems faster and better, their decisions are more innovative and they can even predict the future. The network effect refers to the users' ability to add value by enriching the Web through some specific knowledge and benefiting from the knowledge already available in the Web community.

Figure 10.1 summarizes the relevant aspects of Web 2.0 and provides some examples of Web 2.0 services. The services flickr¹⁴⁵ and del.icio.us¹⁴⁶ illustrate the concept of tagging and collective categorization. In flickr that serves as a place for sharing pictures, every user is able to describe his/her photos by providing a set of own tags. In the services Amazon or eBay¹⁴⁷ users' reviews or evaluations help other users to make purchasing decisions or to assess the credibility of a seller. Wikipedia¹⁴⁸ is a Web encyclopedia to which practically everyone may add content and other users may improve or change it. Wikipedia is based on the trust and collective intelligence that would make it better than any other encyclopedia.

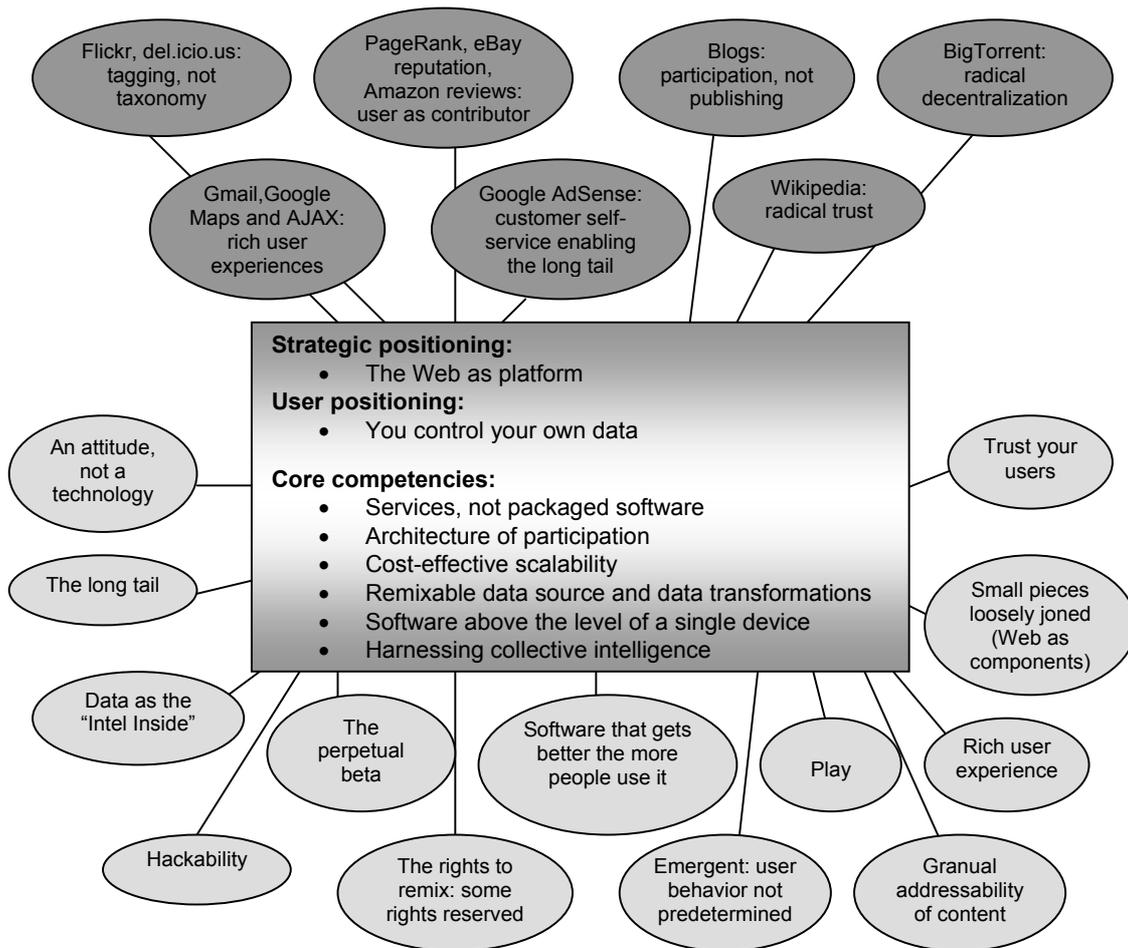


Figure 10.1: Web 2.0 [ORei05]

Mobile Web 2.0 is strictly associated with the idea of Web 2.0. Unlike the programs categorized as Mobile Web 1.0, Mobile Web 2.0 applications are available for download free of charge and are not limited to certain operators. Mobile Web 2.0 extends Web 2.0 services to mobile platforms and takes

¹⁴⁵ <http://www.flickr.com/>.

¹⁴⁶ <http://del.icio.us/>.

¹⁴⁷ <http://www.amazon.com>, <http://www.ebay.com>.

¹⁴⁸ <http://www.wikipedia.org/>.

into account the specifics of mobile environment and mobile users. This implies changes in the methods of content creation and consumption. For example, instead of simply using services through wireless devices users can create content with their help, instant messaging tools take the place of SMS, operators' portals are replaced by mashups created from available services.

Mobile Web 2.0 is supposed to benefit from mobile users' engagement and the opportunity for social interactions. The collective intelligence should refer to the peculiarities of being mobile (e.g. location-based information). The knowledge of mobile users should be collected, other mobile users should be able to comment on the provided information and the data should be easily accessible from mobile devices. There exist already some mobile mashups¹⁴⁹ that integrate content from several sources into one, innovative wireless application. Examples include the Mobile Gmaps service (<http://www.mgmaps.com>) based on Google Maps, Yahoo! Maps, Ask.com Maps and satellite images displayed on Java ME devices or the Socialight (<http://www.socialight.com>) application that allows users to create virtual sticky notes associated e.g. with certain places and to share them with a group of users. The vision of Mobile 2.0 is, however, still far from the reality but this situation may change with the introduction and wide acceptance of 4G technologies.

The user experience on a client (Web-based or mobile) is also changing due to AJAX [CrPaJa05]. AJAX stands for Asynchronous JavaScript and XML and is a method of sending and receiving data (usually in the XML format) from a server-side application through Java Script. AJAX enables the creation of better User Interfaces and provides a standardized method for data retrieval. It includes:

- presentation based on XHTML and CSS
- dynamic display and interaction using the Document Object Model (DOM)
- data interchange/manipulation using XML and XSLT
- asynchronous data retrieval with the help of XMLHttpRequest
- JavaScript as the underlying technology.

By introducing an additional layer (an AJAX engine) between the user and the server the interaction with the server can significantly be improved. At the beginning of a session the browser loads the AJAX engine that is responsible for UI rendering and interactions with the server. The interactions are asynchronous, the user does not have to wait for responses from the server to see the GUI. Actions that do not require communication with the server such as data validation, editing data in memory, or navigations are handled by the AJAX engine. Figure 10.2 shows the difference between traditional Web application model and AJAX model.

¹⁴⁹ Mashup is defined as a Web page or Web applications that seamlessly integrates content from many sources into an integrated experience [cf. http://en.wikipedia.org/wiki/Mashup_%28web_application_hybrid%29].

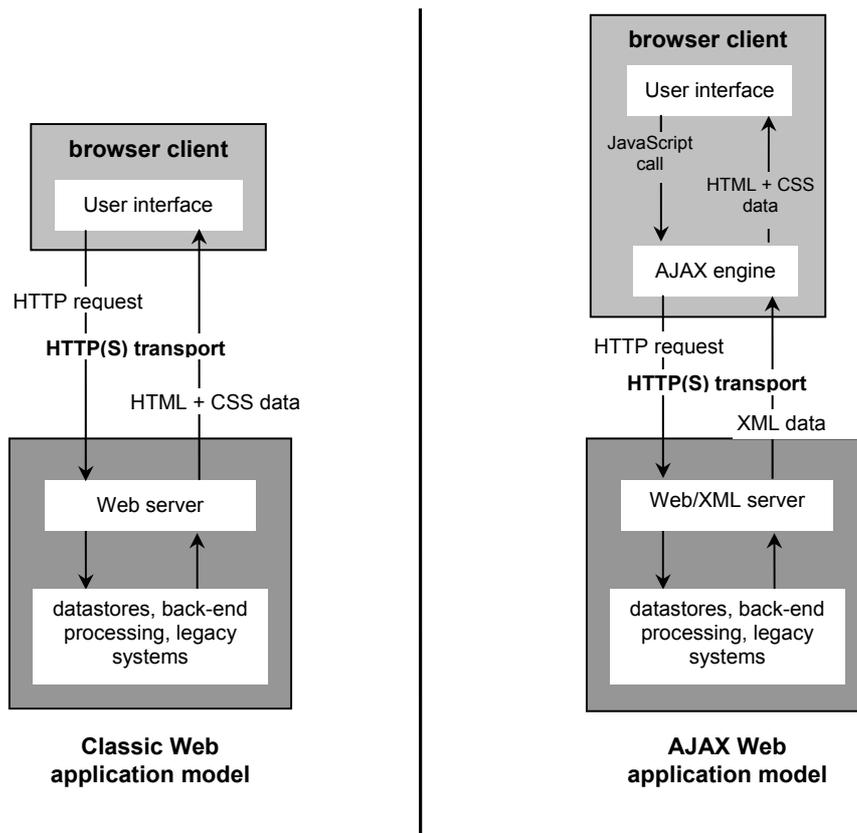


Figure 10.2: Traditional Web application model and Ajax Web application model [cf. CrPaJa05]

AJAX has the potential to replace XHTML and Java ME in the near future. It is already applied in the Opera Platform that runs on mobile devices [Oper06]. AJAX provides a solution for some features of mobile devices and networks that slow down the adoption of mobile services – limited network capacity and restricted input methods. An AJAX application stored on the mobile phone communicates with the help of XML with a Web-server only in a limited number of cases that leads to a decreased need for bandwidth.

The power of the AJAX engine for Opera is also used by SoonR¹⁵⁰. SoonR is a service that allows accessing desktop computers and data stored on them. It is possible to search for certain documents, open files (even if a mobile device does not support the formats like .doc or .pdf), use Microsoft Outlook for reading emails or scheduling, view photos or pictures, benefit from the Skype¹⁵¹ functionality and call Skype users. SoonR is available for download for free and is currently in the Beta stage. There is no need to download any software to a mobile phone, the access to remote computers is browser-based. The user needs the secure SoonR client running on a PC with the connection to the Internet. In order to access the data, the user has to log into his/her account on the SoonR's page. It is possible to use more than one computer for data access but the SoonR clients have to run on all of

¹⁵⁰ <http://www.soonr.com/web/front/ajax.jsp>.

¹⁵¹ <http://www.skype.com>.

them with the same user account. The information displayed on a mobile phone is not in its original format but is transformed by a SoonR's transcoding engine that takes into account the device characteristics.

Although the number of mobile handsets exceeds the number of PCs, the capacities of wireless devices are still far behind those of stationary computers or laptops. As a consequence, critical information is still not stored on mobile devices that offer the "anytime, anywhere" access but on hard disks of computers. The market opportunity that arises from the combination of networked PCs connected to the Internet and mobile devices and services, was estimated by SoonR to be as high as \$2 billion a year¹⁵². Even if the forecasts are totally inadequate, the potential of Mobile Web 2.0 cannot be ignored, especially in the conjunction with the increasing possibilities of 3G and 4G networks. They enable applications that provide rich user experience close to native PC-based programs, can be accessed from all device types without the need to install any additional software on the mobile client or to take up scarce mobile resources and can benefit from AJAX to manage data, reduce latency times and save bandwidth. The critical aspect in the adoption of Mobile Web 2.0 services is, however, the cost of network connections. New pricing strategies such as the introduction of flat rates are needed to encourage the users to benefit from the possibilities of the new generation of applications.

¹⁵² http://www.soonr.com/web/front/images/company_overview.zip

Bibliography

- [3GAm04] 3G Americas: The Evolution of UMTS - 3GPP Release 5 and Beyond, 2004. http://www.3gamericas.org/pdfs/umtsrel5_beyond_june2004.pdf. Download: 2005-10-22.
- [3GPP00] 3rd Generation Partnership Project (3GPP): Architecture Principles for Release 2000, Technical Report, 2000. http://www.3gpp.org/ftp/Specs/2000-06/R2000/23_series/23821-101.zip. Download: 2005-09-22.
- [3GPP05] 3rd Generation Partnership Project (3GPP): 3GPP Specification Series for Mobile Multimedia, 2005. <http://www.3gpp.org/ftp/Specs/html-info/26-series.htm>. Download: 2005-12-02.
- [3GPP05a] 3rd Generation Partnership Project 2: 3GPP2 Specifications, 2005. http://www.3gpp2.org/Public_html/specs/alltsgscfm.cfm. Download: 2005-12-01.
- [3GPP99] 3rd Generation Partnership Project (3GPP): QoS Concept and Architecture, 1999. http://www.3gpp.org/ftp/tsg_ran/WG3_Iu/TSGR3_08/Docs/Pdfs/R3-99e36.pdf. Download: 2005-09-22.
- [Abbo01] Abbott, K.: Voice Enabling Web Applications: VoiceXML and Beyond, APress, Berkeley, 2001.
- [AbPh99] Abrams, M., Phanouriou, C.: UIML: An XML Language for Building Device-Independent User Interfaces; in: *Proceedings of the XML '99 Conference*, Philadelphia, 1999. <http://www.harmonia.com/resources/papers/xml99Final.pdf>. Download: 2005-09-21.
- [Abr+99] Abrams, M. et al.: UIML: An Appliance-Independent XML User Interface Language; in: *Proceedings of the 8th International World Wide Web Conference*, Toronto, 1999, pp. 1695-1708.
- [Abra+03] Abran, A. et al.: Usability Meanings and Interpretations in ISO Standards, in: *Software Quality Journal*, 11 (4), 2003, pp. 325-338.
- [Abra00] Abrams, M.: User Interface Markup Language (UIML) Draft Specification, 2000. <http://www.uiml.org/specs/docs/uiml20-17Jan00.pdf>. Download: 2005-09-12.
- [Abra00a] Abrams, M.: Device-Independent Authoring with UIML; in: *Proceedings of the W3C Workshop on Web Device Independent Authoring*, Bristol, 2000. <http://www.harmonia.com/resources/papers/devindauthoring.htm>. Download: 2005-09-21.
- [AhBa02] Ahonen, T., Barrett, J.: Services for UMTS, John Wiley & Sons, Padstow, 2002.
- [AjFi80] Ajzen, I., Fishbein, M.: Understanding Attitudes and Predicting Social Behavior, Prentice Hall, New Jersey, 1980.
- [Ajze91] Ajzen, I.: The Theory of Planned Behavior; in: *Organizational Behavior and Human Decision Processes*, 50/1991, pp. 179-211.
- [Alam+03] Alam, H. et al.: Web Page Summarization for Handheld Devices: A Natural Language Approach; in: *Proceedings of the 7th Conference on Document Analysis and Recognition*, Edinburgh, 2003, pp. 1153-1157.

- [Albi03]** Albin, S.: *The Art of Software Architecture: Design Methods and Techniques*, Wiley Publishing, Indianapolis, 2003.
- [AlPeAb04]** Ali, M., Pérez-Quiñones, M., Abrams, M.: Building Multi-Platform User Interfaces with UIML; in: Seffah, A., Javahery, H. (Eds.): *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, John Wiley & Sons, Sussex, 2004, pp. 95-118.
- [AlRa03]** Alam, H., Rahman, F.: Web Document Manipulation for Small Screen Devices: A Review; in: *Proceedings of the 2nd International Workshop on Web Document Analysis*, Edinburgh, 2003. http://www.csc.liv.ac.uk/~wda2003/Papers/Section_II/Paper_8.pdf. Download: 2005-09-28.
- [Ambr06]** Ambrosio, C.: *Global Handset Sales Forecast 2005-2010*, 2006. <http://www.strategyanalytics.net/default.aspx?mod=ReportAbstractViewer&a0=2744>. Download: 2006-01-20.
- [AnBaSh01]** Anastopoulou, S., Baber, C., Sharples, M.: Multimedia and Multimodal Systems: Commonalities and Differences; in: *Proceedings of the 5th Human Centered Technology Postgraduate Workshop*, Sussex, 2001. <http://www.cogs.susx.ac.uk/lab/hct/hctw2001/papers/anastopoulou.pdf>. Download: 2005-11-29.
- [Ande01]** Andersson, C.: *GPRS and 3G Wireless Applications: Professional Developer's Guide*, John Wiley & Sons, New York, 2001.
- [AnDi02]** Anckar, B., D'Incau, D.: Value-Added Services in Mobile Commerce: An Analytical Framework and Empirical Findings from a National Consumer Survey; in: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, 2002. <http://csdl.computer.org/comp/proceedings/hicss/2002/1435/03/14350086b.pdf>. Download: 2005-06-12.
- [Andr05]** Andrejko, A.: Improving Adaptive Hypermedia by Adding Semantics; in: *Proceedings of IIT.SRC 2005: Student Research Conference in Informatics and Information Technologies*, Bratislava, 2005, pp. 235-241.
- [Ane+04]** Aneegg, H. et al.: Multimodal Interfaces in Mobile Devices - The MONA Project; in: *Proceedings of the Workshop on Emerging Applications for Wireless and Mobile Access*, New York, 2004. http://mona.ftw.at/papers/MobEAll-Paper_6.pdf. Download: 2005-08-10.
- [AppI04]** Apple Computer: *QuickTime Fundamentals*, 2004, http://developer.apple.com/referencelibrary/API_Fundamentals/QuickTime-fund-date.html. Download: 2005-12-02.
- [ASF05]** Apache Software Foundation: *Batik SVG Toolkit 1.5.1.*, 2005. <http://xml.apache.org/batik>. Download: 2005-01-10.
- [ASF05a]** Apache Software Foundation: *Struts Web Application Framework*, 2005. <http://struts.apache.org>. Download: 2005-01-21.
- [ASF05b]** Apache Software Foundation: *Tomcat Web Container*, 2005. <http://jakarta.apache.org/tomcat/index.html>. Download: 2005-01-21.

- [ASF05c] Apache Software Foundation: Xalan, Version 2.6.0., 2005. <http://xml.apache.org/xalan-j/>. Download: 2005-10-10.
- [ASF05d] Apache Software Foundation: Xerces, v. 1.4.4, 2005. <http://xml.apache.org/xerces-j/>. Download: 2005-10-10.
- [ASFI05] Apache Software Foundation Incubator Projects: MyFaces: Open Source JSF Implementation, 2005. <http://incubator.apache.org/myfaces/index.html>. Download: 2005-01-11.
- [ATKe+03] A.T.Kearney, University of Cambridge: The New Mobile Mindset, 2003. http://www.atkearney.com/shared_res/pdf/Mobinet_Monograph_S.pdf. Download: 2005-12-20.
- [ATKe+04] A.T.Kearney, University of Cambridge: Mobinet Index 2004, 2004. http://www.atkearney.com/shared_res/pdf/Mobinet_Extracts_2004_S.pdf. Download: 2005-12-20.
- [ATKe+05] A.T.Kearney, University of Cambridge: Mobinet 2005, 2005. http://www.atkearney.com/shared_res/pdf/Mobinet_2005_Detailed_Results.pdf. Download: 2006-01-12.
- [AzMeRo00] Azevedo, P., Merrick, R., Roberts, D.: OVID to AUIML - User-Oriented Interface Modeling, in: *Proceedings of the 1st Workshop Towards a UML Profile for Interactive Systems Development (TUPIS'00)*, York, 2000. <http://math.uma.pt/tupis00/submissions/azevedoroberts/azevedoroberts.html>. Download: 2005-01-02.
- [BaCIKa03] Bass, L., Clements, P., Kazman, R.: *Software Architecture in Practice*, 2nd Edition, Addison-Wesley, Boston, 2003.
- [BaJe06] Balik, M., Jelinek, I.: Modeling of Adaptive Hypermedia Systems; in: *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'06)*, Veliko Tarnovo, 2006. <http://ecet.ecs.ru.acad.bg/cst06/Docs/cp/sV/V.8.pdf>. Download: 2006-03-04.
- [Bana+00] Banavar, G. et al.: Challenges: An Application Model for Pervasive Computing; in: *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, Boston, 2000, pp. 266-274.
- [Bana+04] Banavar, G. et al.: An Authoring Technology for Multi-Device Web Applications; in: *IEEE Pervasive Computing*, 3 (3), 2004, pp. 83-93.
- [Bana+04a] Banavar, G. et al.: Tooling and Systems Support for Authoring Multi-Device Applications; in: *Journal of Systems and Software*, 69 (3), 2004, pp. 227-242.
- [Baso05] Basole, R.: Transforming Enterprises through Mobile Applications: A Multi-Phase Framework; in: *Proceedings of the 11th Americas Conference on Information Systems (AMCIS)*, Omaha, 2005. <http://www2.isye.gatech.edu/~rbasole/docs/amcis05.pdf>. Download: 2005-07-02.
- [Baud+04] Baudisch, P. et al.: Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content; in: *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, Santa Fe, 2004, pp. 91-94.

- [Baye03] Bayern, S.: JSTL in Action, Manning Publications, Greenwich, 2003.
- [BeCzRo02] Bederson, B., Czerwinski, M., Robertson, G.: A Fisheye Calendar Interface for PDA: Providing Overviews for Small Displays, University of Maryland HCIL Technical Report #HCIL-2002-09, 2002. <http://hcil.cs.umd.edu/trs/2002-09/2002-09.html>. Download: 2005-02-01.
- [Bede00] Bederson, B.: Fisheye Menu Selection; in: *Proceedings of the ACM User Interface Software and Technology Conference (UIST '00)*, San Diego, 2000, pp. 217-225.
- [BeHo94] Benderson, B., Hollan, J.: PAD++: A Zooming Graphical Interface for Exploring Alternate Interface Physics; in: *Proceedings of the ACM User Interface Software and Technology Conference (UIST '94)*, Marina Del Ray, 1994, pp. 17-26.
- [BeMe98] Bederson, M., Meyer, J.: Implementing a Zooming User Interface: Experience Building Pad++; in: *Journal of Software - Practice and Experience*, 28 (10), 1998, pp. 1101-1135.
- [Benk02] Benkler, Y.: Coase's Penguin, or Linux and The Nature of the Firm; in: *The Yale Law Journal*, 112/2002, 2002, pp. 369-446. <http://www.yale.edu/yalelj/112/BenklerWEB.pdf>. Download: 2006-05-30.
- [BenM03] BenMoussa, Ch.: Workers on the Move: New Opportunities through Mobile Commerce; in: *Proceedings of the IADIS International Conference WWW/Internet 2003*, Lisbon, 2003. <http://web.hhs.se/cic/roundtable2003/papers/d34benmoussa.pdf>. Download: 2005-02-12.
- [Berg+02] Bergman, L.D. et al.: Combining Handcrafting and Automatic Generation of User-Interfaces for Pervasive Devices; in: *Proceedings of Computer-Aided Design of User Interfaces III*, Valenciennes, 2002, pp. 155-166.
- [Berg+04] Van den Bergh, J. et al.: Evaluation of High-Level User Interface Description Languages for Use on Mobile and Embedded Devices; in: *Proceedings of the Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages*, Gallipoli, 2004.
- [Berg04] Bergsten. H.: JavaServer Faces, O'Reilly & Associates, Beijing, 2004.
- [BeRiSt03] Benz, A., Ritz, T., Stender, M.: Marktstudie Mobile CRM-Systeme, Fraunhofer IRB Verlag, Stuttgart, 2003.
- [BeWa03] Beck, J., Wade, M.: DoCoMo – Japan's Wireless Tsunami, Amacom Ltd., New York, 2003.
- [Bian+01] Biancheri, Ch. et al.: EIHA?!?: Deploying Web and WAP Services Using XML Technology; in: *SIGMOD Record*, 30 (1), 2001, pp. 5-12.
- [BiGiSu99] Bickmore, T., Girgensohn, A., Sullivan, J.W.: Web Page Filtering and Re-Authoring for Mobile Users; in: *The Computer Journal*, 42 (6), 1999, pp. 534-546.
- [BiSc97] Bickmore, T.W., Schilit, B.N.: Digester: Device-Independent Access to the World Wide Web; in: *Proceedings of the 6th International World Wide Web Conference*, Santa Clara, 1997, pp. 1075-1082.

- [Björ+00] Björk, S. et al.: PowerView: Structured Access to Integrated Information on Small Screens; in: *Extended Abstracts of CHI'2000*, The Hague, pp. 265-266.
- [Björ+02] Björk, S. et al.: Power View. Using Information Links and Information Views to Navigate and Visualize Information on Small Displays; in: *Proceedings of the 2nd International Symposium of Handheld and Ubiquitous Computing*, Bristol, 2000, pp. 46-62.
- [Björ+99] Björk, S. et al.: WEST: A Web Browser for Small Terminals; in: *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, Asheville, 1999, pp. 187-196.
- [BjRe99] Björk, S., Redström, J.: An Alternative to Scrollbars on Small Screens; in: *Proceedings of the Conference on Human Factors in Computing Systems*, Pittsburgh, 1999, pp. 316-317.
- [BICoDa05] Blechar, J., Constantiou, I.D., Damsgaard, J.: Seeking Answers to the Advanced Mobile Services Paradox: Minimal Acceptance and Use Despite Accessibility; in: *Proceedings of the IFIP International Working Conference on Mobile Information Systems*, Leeds, 2005, pp. 311-318.
- [BIS97] Blair, G., Stefani, J.: *Open Distributed Processing and Multimedia*, Addison-Wesley, Boston, 1997.
- [BoHoSc90] Böcker, H.-D., Hohl, H., Schwab, T.: Hypadapter - Individualizing Hypertext; in: *Proceedings of the IFIP 3rd International Conference on Human-Computer Interaction*, Amsterdam, 1990, pp. 931-936.
- [BoKaMi04] Bouras, C., Kapoulas, V., Misedakis, I.: A Web Page Fragmentation Technique for Personalized Browsing; in: *Proceedings of the 2004 ACM Symposium on Applied Computing*, Nicosia, 2004, pp. 1146-1147.
- [Bov05] Bovens, A.: *Mobile Web Development in Japan: A Tag Soup Tale*, 2005. <http://andreas.web-graphics.com/mobile/2005>. Download: 2006-05-12.
- [BoVa02] Bouillon, L., Vanderdonckt, J.: Retargeting Web Pages to Other Computing Platforms with VAQUITA; in: *Proceedings of Working Conference on Reverse Engineering*, Richmond, 2002, pp. 339-348.
- [BoVa03] Bouillon, L., Vanderdonckt, J.: User Interface Reverse Engineering; in: *Proceedings of the 2nd International Conference on Universal Access in Human-Computer Interaction*, Crete, 2003, pp. 1509-1513.
- [BoVeCh04] Bouillon L., Vanderdonckt J., Chieu Chow K.: Flexible Re-engineering of Web Sites; in: *Proceedings of the International Conference on Intelligent User Interfaces*, Madeira, 2004, pp. 132-139.
- [Bran98] Brandenburg, K.: Perceptual Coding of High Quality Digital Audio; in: Kahrs, M., Brandenburg, K. (Eds.): *Applications of Digital Signal Processing to Audio and Acoustics*, Kluwer Academic, New York, 1998, pp. 39-83.
- [BrCrGa02] Brodgen, B., D'Cruz, C., Gaither, M.: *Cocoon 2 Programming: Web Publishing with XML and Java*, Sybex, Alameda, 2002.

- [Bria+01] Bria, A. et al.: 4th Generation Wireless Infrastructures: Scenarios and Research Challenges; in: *IEEE Personal Communications*, 8 (6), 2001, pp. 25 -31.
- [Brit00] Britton, Ch.: *IT Architectures and Middleware: Strategies for Building Large, Integrated Systems*, Addison-Wesley: Boston, 2000.
- [BrPeZy93] Brusilovsky, P., Pesin, L., Zyryanov, M.: Towards an Adaptive Hypermedia Component for an Intelligent Learning Environment; in: *Proceedings of 3rd East-West International Conference on Human-Computer Interaction (EWHCI '93)*, Moscow, 1993, pp. 348-358.
- [Brus01] Brusilovsky, P.: Adaptive Hypermedia; in: *User Modeling and User-Adapted Interaction*, 11 (1), 2001, pp. 87-110.
- [Brus03] Brusilovsky, P.: Adaptive Navigation Support in Educational Hypermedia: the Role of the Student Knowledge Level and the Case for Meta-Adaptation; in: *British Journal of Educational Technology*, 34 (4), 2003, pp. 487-497.
- [Buch+01] Buchanan, G. et al.: Improving Mobile Internet Usability; in: *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, 2001, pp. 673-680.
- [Buch+02] Buchholz, S. et al.: Software Architecture for the Adaptation of Dialogs and Contents to Different Devices; in: *Proceedings of the International Conference on Information Networking, Wireless Communications Technologies and Network Applications*, Cheju Island, 2002, pp. 42-51
- [BuMoPa00] Buyukkokten, O., Garcia-Molina, H., Paepcke, A.: Focused Web Searching with PDAs; in: *Proceedings of the 9th International World Wide Web Conference on Computer Networks*, Amsterdam, 2000, pp. 213-230.
- [BuMoPa01] Buyukkokten, O., Garcia-Molina, H., Paepcke, A.: Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices; in: *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, 2001, pp. 652-662.
- [BuPaMo01] Buyukkokten, O., Paepcke, A., Garcia-Molina, H.: Text Summarization of Web Pages on Handheld Devices; in: *Proceedings of the Workshop on Automatic Summarization (NAACL 2001)*, Pittsburgh, 2001. <http://www.isi.edu/~cyl/was-naacl2001/papers/orkut-p1.pdf>. Download: 2005-09-25.
- [Butl+01] Butler, M.: Current Technologies for Device Independence, Hewlett Packard Working Paper, HPL-2001-83, 2001. <http://www.hpl.hp.com/techreports/2001/HPL-2001-83.html>. Download: 2005-09-15.
- [Butl+02] Butler, M. et al.: Device Independence and the Web; in: *IEEE Internet Computing*, 6 (5), 2002, pp. 81-86.
- [Butl02] Butler, M.: CC/PP and UAProf: Issues, Improvements and Future Directions; in: *Proceedings of the W3C Delivery Context Workshop*, Sophia-Antipolis, 2002. <http://www.hpl.hp.com/techreports/2002/HPL-2002-35.pdf>. Download: 2005-10-29.
- [Butl02a] Butler, M.: DELI: A Delivery Context Library for CC/PP and UAProf, External Technical Report HPL-2001-260, 2002. <http://www.hpl.hp.com/personal/marbut/DeliUserGuideWEB.htm>. Download: 2005-10-30.

- [Buyu+00] Buyukkokten, O. et al.: Power Browser: Efficient Web Browsing for PDAs; in: *Proceedings of the ACM Conference on Computers and Human Interaction 2000 (CHI'00)*, The Hague, 2000, pp. 430-437.
- [CaMaSc99] Card, S.K., MacKinlay, J.D., Schneiderman, B.: *Readings in Information Visualization - Using Vision to Think*, Morgan Kaufman, San Francisco, 1999.
- [Cann03] Cannistrà, F.: SADiC: The Semantic API for the Delivery Context. The SADiC Semantic Approach, 2003. <http://www.the-web-middle-earth.com/sadic/sadicSemantics.html>. Download: 2005-10-29.
- [Cant+06] Cantera, J., Jiménez, M., López, G., Soriano, J.: Using the Semantic Web in Ubiquitous and Mobile Computing: the Morfeo Experience; in: *International Journal of Computer Science*, 1 (2), 2006, pp. 108-126.
- [Carb+02] Carboni, D. et al.: Interactions Model and Code Generation for J2ME Applications; in: *Proceedings of the 4th Symposium on Mobile Human-Computer Interaction*, Pisa, 2002, pp. 286-290.
- [Carb+02a] Carboni, D.: E-MATE: An Open Architecture to Support Mobility of Users; in: *Proceedings of the 5th International Baltic Conference on Databases and Information Systems*, Tallin, 2002, pp. 227-242.
- [Carb+03] Carboni, D. et al.: Filling the Gap between Users and Objects: a Multichannel Interactive Environment; in: *Proceedings of the 4th AI*IA/TABOO Joint Workshop "From Objects to Agents": Intelligent Systems and Pervasive Computing (WOA'03)*, Villasimius, 2003, pp.187-190.
- [CaRo02] Cade, M., Roberts, S.: *Sun Certified Enterprise Architect for J2EE™ Technology Study Guide*, Prentice Hall, London, 2002.
- [Cast+04] Castells, M. et al.: *The Mobile Communication Society: A Cross-Cultural Analysis of Available Evidence on the Social Uses of Wireless Communication Technology*, Annenberg Research Report, 2004. <http://arnic.info/WirelessWorkshop/MCS.pdf>. Download: 2006-01-01.
- [Cava04] Cavaness, Ch.: *Programming Jakarta Struts*, 2nd Edition, O'Reilly & Associates, Beijing, 2004.
- [CaWa02] Carlsson, C., Walden, P.: Further Quests for Value-Added Products & Services in Mobile Commerce; in: *Proceedings of the European Conference on Information Systems (ECIS 2002)*, Gdansk, 2002, pp. 715-724.
- [CaWa02a] Carlsson, C., Walden, P.: Mobile Commerce: A Summary of Quests for Value-Added Products & Services; in: *Proceedings of the 15th Bled Electronic Commerce Conference, e-Reality: Constructing the e-Economy*, Bled, 2002, pp. 463-476.
- [CDMA04] CDMA Development Group: *Using Wireless to Provide Universal Access to Telecom Services*, 2004. http://www.cdg.org/resources/white_papers/files/Universal_Services_10-28-04.pdf. Download: 2005-11-23.

- [CeKuNo03]** Ceska, T., Kuhlins, S., Nösekabel, H.: Evaluation of Wireless Usability with a Java Software Agent; in: *Proceedings of the 7th IASTED International Conference on Internet and Multimedia Systems and Applications*, Honolulu, 2003, pp. 745-750.
- [CeKuNo04]** Ceska, T., Kuhlins, S., Nösekabel, H.: Programmgestützte Analyse von WAP-Sites; in: *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI04)*, Essen, 2004, Band 3, pp. 54-64.
- [Cera02]** Cerami, E.: *Web Services Essentials. Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*, 1st Edition, O'Reilly & Associates, Beijing, 2002.
- [ChaKi04]** Chae, M., Kim, J.: Size and Structure Matter to Mobile Users: An Empirical Study of the Effects of Screen Size, Information Structure, and Task Complexity on User Activities with Standard Web Phones; in: *Behaviour & Information Technology*, 23 (3), 2004, pp. 165-181.
- [ChEI99]** Chandra, S., Ellis, C.S.: JPEG Compression Metric as a Quality Aware Image Transcoding; in: *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, Berkeley, 1999, pp. 81-92.
- [ChEIVa00]** Chandra, S., Ellis, C.S., Vahdat, A.: Application-Level Differentiated Multimedia Web Services Using Quality Aware Transcoding; in: *IEEE Journal on Selected Areas in Communications*, 18 (12), 2000, pp. 2544-2565.
- [ChEIVa99]** Chandra, S., Ellis, C.S., Vahdat, A.: Multimedia Web Services for Mobile Clients Using Quality Aware Transcoding; in: *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Multimedia (WoWMoM 99)*, Seattle, 1999, pp. 99-108.
- [Chki03]** Chae, M., Kim, J.: What's so Different about Mobile Internet; in: *Communications of the ACM*, 46 (12), 2003, pp. 240-247.
- [ChKo00]** Chen, G., Kotz, D.: A Survey of Context-Aware Mobile Computing Research, Dartmouth College Technical Report: TR2000-381, 2000. <ftp://ftp.cs.dartmouth.edu/TR/TR2000-381.pdf>. Download: 2005-11-10.
- [ChMaZh03]** Chen, Y., Ma, W., Zhang, H.: Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices; in: *Proceedings of the 12th International Conference on World Wide Web*, Budapest, 2003, pp. 225-233.
- [ChScSc02]** Christoffel, M., Schmitt, B., Schneider, J.: Semi-Automatic Wrapper Generation and Adaptation; in: *Proceedings of the 4th International Conference on Enterprise Information Systems*, Ciudad Real, 2002, pp. 60-67
- [Clar01]** Clarke, I.: Emerging Value Propositions for M-Commerce; in: *Journal of Business Strategies*, 18 (2), 2001, pp. 133-148.
- [Clas02]** Classen, M.: XParse Parser, 2002. <http://webreference.com/xml/tools>. Download: 2005-10-10.
- [CoBIDa05]** Constantiou, I.D., Blechar, J., Damsgaard, J.: New Mobile Services and the Internet: Friend or Foe?; in: *Proceedings of the Conference of the Information Systems Research*

- in Scandinavia Association (IRIS28)*, Kristiansand, 2005. <http://www.hia.no/iris28/Docs/IRIS2028-1041.pdf>. Download: 2006-01-02.
- [CoLo99]** Constantine, L., Lockwood, L.: *Software for Use: A Practical Guide to the Models and Methods of Usage-Centred Design*, Addison-Wesley, New York, 1999.
- [CoLoSa03]** Cockburn, A., Looser, J., Savage, J.: Around the World in Seconds with Speed-Dependent Automatic Zooming; in: *Proceedings of the ACM User Interface Software and Technology*, Vancouver, 2003, pp. 35-36.
- [Coni+04]** Coninx, K. et al.: Building User Interfaces with Tasks, Dialogs and XML; in: *Proceedings of the ACM International Conference on Intelligent User Interfaces and Computer Aided Design of User Interfaces (IUI/CADUI'2004)*, Funchal, 2004. <http://research.edm.luc.ac.be/kris/research/publications/cadui2004/cadui2004-demo.pdf>. Download: 2005-12-29.
- [Cons03]** Consensus: Adaptation Engine: Installation and Configuration Guide, 2003. <http://www.consensus-online.org/publicdocs/D7public.pdf>. Download: 2005-10-10.
- [Cons04]** Consensus: RIML Language Specification, Version 2, 2004. <http://www.consensus-online.org/publicdocs/20040317-RIML-II-Final-public.pdf>. Download: 2005-12-01.
- [Cors01]** Corston-Oliver, S.: Text Compaction for Display on Very Small Screens; in: *Proceedings of the Workshop on Automatic Summarization (NAACL 2001)*, Pittsburgh, 2001, pp. 89-98.
- [CoSm01]** Collins, D., Smith, C.: *3G Wireless Networks*, McGraw-Hill Professional, New York, 2001.
- [Cout+03]** Coutts, P., Alport, K., Coutts, R., Morell, D.: Beyond The Wireless Internet Hype - Re-engaging the User; in: *Proceedings of the Communications Research Forum (CRF'03)*, Canberra, 2003. [http://www.smartinternet.com.au/SITWEB/publication/files/80_\\$\\$\\$_91432/P04_038_paper.pdf](http://www.smartinternet.com.au/SITWEB/publication/files/80_$$$_91432/P04_038_paper.pdf). Download: 2005-12-20.
- [Covi05]** Covington, R.: The Year of The (Mo)Blogger; in: *Receiver*, 12/2005. http://www.receiver.vodafone.com/12/articles/pdf/12_08.pdf. Download: 2006-01-01.
- [CrLa02]** Crnkovic, I., Larsson, M. (Eds.): *Building Reliable Component-Based Software Systems*, Artech House Publishers, Norwood, 2002.
- [CrPaJa05]** Crane, D., Pascarello, E., James, D.: *Ajax in Action*, Manning Publications, Greenwich, 2005.
- [Curb+02]** Curbera, F. et al.: Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI; in: *IEEE Internet Computing*, March/April 2002, pp. 86-93.
- [DaGrJo05]** Davis, V., Gray, J., Jones, J.: Generative Approaches for Application Tailoring of Mobile Devices; *Proceedings of the 43rd Annual ACM SE Conference*, Kennesaw, 2005. <http://www.cis.uab.edu/gray/Pubs/acmse-2005.pdf>. Download: 2005-01-12.

- [Dams+05]** Damsgaard, J. et al.: Seeking Answers to the Advanced Mobile Services Paradox: Minimal Acceptance and Use Despite Accessibility; in: *Proceedings of IFIP TC8 Working Conference on Mobile Information Systems*, Leeds, 2005, pp. 311-318.
- [Davi89]** Davis, F.D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology; in: *MIS Quarterly*, 13 (3), 1989, pp. 319-339.
- [DaZeGr05]** Daley, E., Zetie, C., Gray, B.: Mobile Application Adoption Leaps Forward: Interest Shifts To Line-Of-Business Applications, 2005. <http://www.forrester.com/Research/Document/Excerpt/0,7211,37250,00.html>. Download: 2006-01-12.
- [Delo05]** Deloitte: TMT Trends: Predictions, 2005. A Focus on the Mobile and Wireless Sector, 2005. http://www.deloitte.com/dtt/cda/doc/content/Mobile%20wireless_FINAL_01FEB05_LR_FA_LOCKED.pdf. Download: 2005-12-10.
- [Derm+03]** Dermler, G. et al.: Flexible Pagination and Layouting for Device Independent Authoring; in: *Proceedings of the Workshop on Emerging Applications for Wireless and Mobile Access*, Budapest, 2003. http://www.research.att.com/~rjana/dermler_wasmund.pdf. Download: 2005-10-10.
- [Dert99]** Dertouzos, M.: The Oxygen Project; in: *Scientific American*, 282 (3), 1999, pp. 52–63.
- [DeSaAb01]** Dey, A.K., Salber, D., Abowd, G.: Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications; in: *Human-Computer Interaction (HCI) Journal*, 16 (2-4), 2001, pp. 97-166.
- [Desm+94]** Desmarais, M.C. et al.: A Survey on User Expectations for Interface Builders; in: *Proceedings of the Conference on Human Factors in Computing Systems*, Boston, 1994, pp. 279 – 280.
- [Dete04]** Detecon: Telco 2010. Emerging Telecommunication Landscapes: the Cards Are Reshuffled, Detecon Consulting Publishing, Bonn, 2004.
- [Dey01]** Dey, A. K.: Understanding and Using Context; in: *Personal and Ubiquitous Computing Journal*, 5 (1), 2001, pp. 4-7.
- [DoSe01]** Donyaee, M., Seffah, A.: QUIM: An Integrated Model for Specifying and Measuring Quality in Use; in: *Proceedings of the 8th IFIP Conference on Human Computer Interaction*, Tokyo, 2001.
- [Drak+05]** Drake, S. et al: Worldwide Mobile Worker Population 2005-2009, Forecast and Analysis, IDC study, October 2005. <http://www.idc.com/getdoc.jsp?containerId=34124>. Download: 2006-01-01.
- [DuBr02]** Dunlop, M., Brewster, S.: The Challenge of Mobile Devices for Human-Computer Interaction; in: *Personal and Ubiquitous Computing*, 6 (4), pp. 235-236.
- [DuLeWi03]** Dudley, B., Lehr, J., Willis, B.: Mastering JavaServer Faces, Wiley Publishing, Indianapolis, 2004.
- [Ecke03]** Eckel, B.: Thinking in Java, 3rd Edition, Prentice Hall, New York, 2003.
- [Econ06]** The Economist: Texting: Hot to Trot; in: *The Economist*, 2006-04-01, p. 62.

- [EdKo04]** Edelmann, J., Koivuniemi, J.: Future Development of Mobile Services and Applications Examined Through the Real Options Approach; in: *Teletronikk*, 2/2004, pp. 48-57.
- [EiVaPu+01]** Eisentein, J., Vanderdonckt, J., Puerta, A.: Applying Model-Based Techniques to the Development of UIs for Mobile Computers; in: *Proceedings of the ACM Intelligent User Interfaces Conference*, Santa Fe, 2001, pp. 69-76.
- [EiVaPu00]** Eisenstein, J., Vanderdonckt, J., Puerta, A.: Adapting to Multiple Contexts with User-Interface Modeling; in: *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, Monterey, 2000, pp. 83-94.
- [EkHoOI01]** Ekudden, E., Horn, U., Olin, J.: On-Demand Mobile Media - A Rich Service Experience for Mobile Users; in: *Ericsson Review*, Vol. 4, 2001, pp. 168-177.
- [EIPh04]** Elliott, G., Phillips, N.: *Mobile Commerce and Wireless Computing Systems*, Pearson Educations Limited, London, 2004.
- [Enge04]** Engelen van, R.: Code Generation Techniques for Developing LightWeight XML Web Services for Embedded Devices; in: *Proceedings of the 9th ACM Symposium on Applied Computing (SAC)*, Nicosia, 2004, pp. 854-861.
- [Enge05]** Engelen van, R.: gSOAP Toolkit, 2005. <http://sourceforge.net/projects/gsoap2>. Download: 2005-12-11.
- [Engl97]** Englander, R.: *Developing Java Beans*, O'Reilly & Associates, Sebastopol, 1997.
- [Enhy04]** Enhydra: kSOAP, Version 2.0, 2004. <http://ksoap.objectweb.org/>. Download 2004-06-23.
- [Enhy04a]** Enhydra: kXML-RPC, 2004. <http://kxmlrpc.objectweb.org/>. Download 2004-06-23.
- [Enhy02]** Enhydra: kXML Parser, 2002. <http://kxml.objectweb.org>. Download: 2005-10-10.
- [Eric02]** Ericsson: Von GSM zu UMTS: Informationen über neue Mobilfunktechniken, 2002. http://www.ericsson.com/de/broschueren/von_gsm_zu_umts.pdf. Download 2005-10-22.
- [Eric03]** Ericsson: EDGE - Introduction of High-Speed Data in GSM/GPRS Networks, 2003. http://www.ericsson.com/products/white_papers_pdf/edge_wp_technical.pdf. Download 2004-10-07.
- [EsSm04]** Eslambolchilar, P., Smith-Murray, R.: Tilt-Based Automatic Zooming and Scaling in Mobile Devices – A State-Space Implementation; in: *Proceedings of the 6th International Conference on Human Computer Interaction with Mobile Devices and Services*, Glasgow, 2004. <http://citeseer.ist.psu.edu/658800.html>. Download: 2005-09-23.
- [Evan02]** Evans, N.: *Business Agility: Strategies for Gaining Competitive Advantage*, Prentice Hall, Upper Saddle River, 2002.
- [Faup06]** Faupel, T.: Hindernisse beim Datenzugriff mittels mobiler Kommunikationstechnologie – eine empirische Analyse; in: *Proceedings of the 1st MMS Conference*, Passau, 2006, pp. 57-67.
- [FAZ05]** Frankfurter Allgemeine Zeitung: T-Mobile kooperiert mit Google bei Internet-Angebot, 2005. <http://www.faz.net/s/RubE2C6E0BCC2F04DD787CDC274993E94C1/Doc~E6D1>

- D6A0006EA4E11B3406C44FDFFEA25~ATpl~Ecommon~Scontent.html. Download: 2006-05-23.
- [Fest02]** Festa, P.: Opera Phone Browser Could Upstage Rivals, 2002, <http://news.com.com/2100-1023-961831.html>. Download: 2005-10-13.
- [FiKoBa02]** Fields, D., Kolb, M., Bayern, S.: *Web Development with JavaServer Pages*, Manning Publications, Greenwich, 2002.
- [FiTa02]** Fielding, R., Taylor, R.: Principled Design of the Modern Web Architecture; in: *ACM Transactions on Internet Technology*, 2 (2), 2002, pp. 115–150.
- [Fort+01]** Forta, B. et al.: *WAP, WML und WMLScript*, Markt+Technik Verlag, München, 2001.
- [Fost+02]** Foster, J. et al.: *Developing Web Services with Java APIs for XML Using WSDP*, Syngress Publishing, Rockland, 2002.
- [Fowl03]** Fowler, M.: *Patterns of Enterprise Application Architecture*, Addison-Wesley, Boston, 2003.
- [Fox+98]** Fox, A. et al.: Experience with Top Gun Wingman, a Proxy-Based Graphical Web Browser for the 3Com PalmPilot; in: *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, The Lake District, 1998, pp. 407-424.
- [Fox+98a]** Fox, A. et al.: Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives, in: *IEEE Personal Communications*, 5 (4), 1998. pp. 10-19.
- [Fox97]** Fox, A.: The Case for TACC: Scalable Servers for Transformation, Aggregation, Caching and Customization, 1997. <http://www.cs.berkeley.edu/~fox/papers/quals.ps>. Download: 2005-09-10.
- [FrKuLi01]** Freire, J., Kumar, B., Lieuwen, D.: WebViews: Accessing Personalized Web Content and Services; in: *Proceedings of the 10th World Wide Web Conference*, Hong Kong, 2001, pp. 576-586.
- [FuBa01]** Funk, J., Barham, S.: *The Mobile Internet: Why Japan Dialed Up and the West Disconnected*, ISI Publications Ltd, New York, 2001.
- [Funk00]** Funk, J.: The Mobile Internet Market: Lessons from Japan's i-mode System; in: *Proceedings of the E-Business Transformation Sector Developments and Policy Implications Conference*, Washington, 2000. http://e-economy.berkeley.edu/conferences/9-2000/EC-conference2000_papers/Funk.pdf. Download: 2005-12-29.
- [Funk03]** Funk, J.: The Origins of New Industries: the Case of Mobile Internet; in: *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET'03)*, Portland, 2003. <http://www.rieb.kobe-u.ac.jp/academic/ra/dp/English/dp134.PDF>. Download: 2005-12-23.
- [Funk04]** Funk, J.: Key Technological Trajectories and the Expansion of Mobile Internet Applications; in: *Journal of Policy, Regulation and Strategy for Telecommunications*, 6 (3), 2004, pp. 208-215.

- [Funk04a]** Funk, J.: The Crisis in the Western Mobile Internet, ITU Telecom Asia, 2004. <http://www.itudaily.com/new/home.asp?articleid=4091101>. Download: 2005-12-24.
- [GaBeLa03]** Gaeremynck, Y., Bergman, L., Lau, T.: MORE for Less: Model Recovery from Visual Interfaces for Multi-Device Application Design; in: *Proceedings of the 8th International Conference on Intelligent User Interfaces*, Miami, 2003, pp. 69-76.
- [GaBuMc03]** Gallardo, D., Burnette, E., McGovern, R.: *Eclipse in Action: A Guide for Web Developers*, Manning Publications, Greenwich, 2003.
- [Gamm+95]** Gamma, E. et al.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Boston, 1995.
- [Gart05]** Gartner: Gartner Outlines Key Trends for Mobile Technology and Subscriber Evolution, 2005. http://www.gartner.com/press_releases/asset_125194_11.html. Download: 2006-01-04.
- [GeHo04]** Geary, D., Horstmann, C.: *Core JavaServer Faces*, Addison-Wesley, Boston, 2004.
- [GeSh04]** Gebauer, J., Shaw, M.: Success Factors and Impacts on Mobile Business Applications: Results from a Mobile E-Procurement Study; in: *International Journal of Electronic Commerce*, 8 (3), 2004, pp. 19-41.
- [Gitm03]** Gitman, M.: Keep up with the Web Service Styles (and Uses); in: *JavaWorld.com*, 2003. <http://www.javaworld.com/javaworld/jw-10-2003/jw-1003-wsstyles.html?page=1>. Download: 2005-10-10.
- [Goeb+01]** Göbel, S. et al.: Device Independent Representation of Web-based Dialog and Contents; in: *Proceedings of the IEEE Youth Forum in Computer Science and Engineering (YUFORIC'01)*, Valencia, 2001. http://www.rn.inf.tu-dresden.de/scripts_Isrn/veroeffent_print/YUFORIC2001.pdf. Download: 2005-09-21.
- [Goog06]** Google: Google SOAP Search API (beta), 2006. <http://www.google.com/apis/>. Download: 2006-05-06.
- [Gosn05]** Gosnell, D.: *Professional Web APIs: Google, eBay, Amazon.com, MapPoint, FedEx*, Wiley Publishing, Indianapolis, 2005.
- [Goth04]** Goth, G.: Critics Say Web Services Need a REST; in: *IEEE Distributed Systems Online*, 5 (12), 2004. <http://csdl2.computer.org/comp/mags/ds/2004/12/oz001.pdf>. Download: 2005-12-12.
- [Gott+02]** Gottschalk, K. et al.: Introduction to Web Services Architecture; in: *IBM System Journal*, 41 (2), pp. 170-171.
- [Goya05]** Goyal, V.: J2ME Tutorial, Part 2: User Interfaces with MIDP 2.0, 2005. <http://today.java.net/pub/a/today/2005/05/03/midletUI.html?page=last>. Download: 2006-04-05.
- [Gras+02]** Grassel, G. et al.: Definition and Prototyping of a Renderer-Independent Markup Language for Enterprise Applications, W3C DIAT Workshop, St. Leon-Rot, 2002. <http://www.w3.org/2002/07/DIAT/posn/nokia-ibm-sap.html>. Download: 2005-08-22.

- [GuFe04] Gutwin, C., Fedak, C.: Interacting with Big Interfaces on Small Screens: a Comparison of Fisheye, Zoom, and Panning Techniques; in: *Proceedings of the Graphics Interface Conference '04*, London, 2004, pp. 145-152.
- [Gupt+03] Gupta, S. et al.: DOM-Based Content Extraction of HTML Documents; in: *Proceedings of the 12th International Conference on World Wide Web*, Budapest, 2003, pp. 207-214.
- [GuSi05] Gulli, A., Signorini, A.: The Indexable Web is More than 11.5 Billion Pages; in: *Proceedings of the WWW 2005 Conference*, Chiba, 2005, pp. 902-903.
- [Gutw02] Gutwin, C.: Improving Focus Targeting in Interactive Fisheye Views; in: *Proceedings of the ACM Conference on Human Factors in Computing System*, Minneapolis, 2002, pp. 267-274.
- [Haar00] Haartsen, J.: The Bluetooth Radio System; in: *IEEE Personal Communications*, 7 (1), 2000, pp. 28-36.
- [HaBuLa04] Haefel, R., Burke, B., Labourey, S.: *Enterprise JavaBeans*, 4th Edition, O'Reilly & Associates, Sebastopol, 2004.
- [Hall01] Hall, M.: *More Servlets and JavaServer Pages*, Prentice Hall, Palo Alto, 2001.
- [Harm02] Harmonia: User Interface Markup Language (UIML) Specification, Version 3.0, 2002. <http://www.uiml.org/specs/>. Download: 2005-12-28.
- [Heat03] Heatton, J.: *JSTL: JSP Standard Tag Library Kick Start*, SAMS Publishing, Indianapolis, 2003.
- [Hein03] Heintze, S.: *VoiceXML – Markt und Möglichkeiten*, XML Clearinghouse Report, Berlin, 2003.
- [Hill01] Hillerson, G.: *Web Clipping Developer's Guide*, 2001. <http://www.palmos.com/dev/tech/docs/>. Download: 2005-10-02.
- [HoAbOn03] Hori, M., Abe, M., Ono, K.: Extensible Framework of Authoring Tools for Web Document Annotation; in: *Proceedings of the International Workshop on Semantic Web Foundations and Application Technologies*, Nara, 2003. <http://www.kasm.nii.ac.jp/SWFAT/PAPERS/SWFAT20R.PDF>. Download: 2005-09-23.
- [Hol92] Holland, J.H.: *Adaptation in Natural and Artificial Systems*, MIT Press, Massachusetts, 1992.
- [HolI03] Hollar, R.: SOAP's Two Messaging Styles; in: *SOA Web Services Journal*, online version, 2003. <http://webservices.sys-con.com/read/39901.htm>. Download: 2006-02-03.
- [Hori+00] Hori, M. et al.: Annotation-Based Web Content Transcoding; in: *Proceedings of the 9th International World Wide Web Conference on Computer Networks*, Amsterdam, 2000, pp. 197-211.
- [Hori02] Hori, M.: Semantic Annotation for Web Content Adaptation; in: Fensel, D. et al. (Eds.): *Spinning the Semantic Web*, MIT Press, Boston, 2002, pp. 542-573.

- [Hors05] Horstmann, T.: Standards als Voraussetzung für erfolgreiche mobile Anwendungen, W3C-Tag, Berlin, 2005. <http://www.w3c.de/Events/2005/W3C-Tag-Presentations/W3C-Tag-2005-Horstmann.pdf>. Download: 2006-01-02.
- [HoTo01] Holma, H., Toskala, A.: WCDMA for UMTS Radio Access for Third Generation Mobile Communications, John Wiley & Sons, New York, 2001.
- [HP05] HP Labs: DELI: A Delivery Context Library For CC/PP and UAProf, 2005. <http://delicon.sourceforge.net/>. Download: 2005-01-02.
- [Hueb+03] Hübsch, G. et al.: Systemlösungen für die Entwicklung adaptiver Anwendungen für mobile und ubiquitäre Infrastrukturen; in: *HMD, Praxis der Wirtschaftsinformatik*, Heft 229, 2003, pp. 42-55.
- [Hueb+05] Hübsch, G. et al.: An Integrated Platform for Mobile, Context-Aware, and Adaptive Enterprise Applications; in: Ferstl, O. et al. (Eds.): *Wirtschaftsinformatik 2005 - eEconomy, eGovernment, eSociety*, Physica-Verlag, Heidelberg, 2005, pp. 1105-1124.
- [Hust+03] Husted, T. et al.: *Struts in Action*, Manning Publications, Greenwich, 2003.
- [HwSeKi02] Hwang, Y., Seo, E., Kim, J.: WebAlchemist: A Structure-Aware Web Transcoding System for Mobile Devices; in: *Proceedings of the Mobile Search Workshop*, Honolulu, 2002. <http://davinci.snu.ac.kr/Download/msw02.pdf>. Download: 2005-09-20.
- [HwSeKi03] Hwang, Y., Seo, E., Kim, J.: Structure-Aware Web Transcoding for Mobile Devices; in: *IEEE Internet Computing*, 7 (5), 2003, pp. 14-21.
- [IBM04] IBM: WebSphere Transcoding Publisher, 2004. <http://www.ibm.com/websphere/transcoding/>. Download: 2005-07-21.
- [IgHi00] Igarashi, T., Hinckely, K.: Automatic Speed-Dependent Zooming for Browsing Large Documents; in: *Proceedings of the 13th Annual Symposium on User Interface Software and Technology*, San Diego, 2000, pp. 139-148.
- [ISO01] International Organization for Standardization: Software Engineering - Product Quality - Part 1: Quality Model (ISO9126-1), 2001.
- [ISO03] International Organisation for Standardization Software Engineering - Product Quality - Part 2: External Metrics (ISO9126-2), 2003.
- [ISO03a] International Organisation for Standardization Software engineering - Product Quality - Part 3: Internal Metrics (ISO9126-3), 2003.
- [ISO04] International Organisation for Standardization Software Engineering - Product Quality - Part 4: Quality in Use Metrics (ISO9126-4), 2004.
- [ISO86] International Organization for Standardization: Information Processing-Text and Office Systems - Standard Generalized Markup Language (SGML), ISO 8879, 1986. <http://www.iso.ch/cate/d16387.html>. Download: 2005-12-20.
- [ISO97] ISO/IEC: The Virtual Reality Modeling Language, 1997. <http://www.web3d.org/x3d/specifications/vrml/vrml97am1/fdam/index.html>. Download: 2005-12-02.

- [ISO98] International Organisation for Standardization: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) - Part 11: Guidance on Usability (ISO9241-11), 1998.
- [ITU02] International Telecommunication Union: Internet for Mobile Generation, 2002. <http://www.itu.int/osg/spu/publications/sales/mobileinternet/>. Download: 2005-10-23.
- [ITU05] International Telecommunication Union: ITU World Telecommunication Indicators Database, 2005. http://www.itu.int/ITU-D/ict/statistics/at_glance/cellular04.pdf. Download: 2005-09-01.
- [Iver04] Iverson, W.: Real World Web Services, O'Reilly & Associates, Sebastopol, 2004.
- [IWS05] Internet World Statistics: Internet Usage, 2005. <http://www.internetworldstats.com/stats.htm>. Download: 2005-12-10.
- [Jaka05] Jakarta Project: Image Tag library 1.0. <http://jakarta.apache.org/taglibs/sandbox/doc/image-doc/intro.html> Download: 2005-10-10.
- [Jama03] Jamalipour, A.: The Wireless Mobile Internet: Architectures, Protocols, and Services, John Wiley & Sons, New York, 2003.
- [Jank03] Jankowska, A.M.: Modeling Contextual Information for Context-Aware Mobile Applications; in: *Proceedings of the Conference on Research of Contemporary Economics Issues*, Lubniewice, 2003, pp. 41-51.
- [Jank03a] Jankowska, B.: Architecture for Integrated Mobile and Desktop-Based Applications; in: *Proceedings of the Conference on Research of Contemporary Economics Issues*, Poznan University of Economics Publishing House, Lubniewice, 2003, pp. 52-61.
- [Jank04] Jankowska, B.: Mobile Interfaces Tag Library - A Framework for Single Source Publishing; in: *Proceedings of the 3rd Conference on Advanced Information Technologies for Management*, Karpacz, 2004, pp. 470-479.
- [Jank05] Jankowska, B.: Approaches for Device-Independent Content Delivery to Mobile Devices; in: Ferstl, O. et al. (Eds.): *Wirtschaftsinformatik 2005 - eEconomy, eGovernment, eSociety*, Physica-Verlag, Heidelberg, 2005, pp. 1561-1580.
- [JaTe01] Jamalipour, A.; Tekinay, S.: Fourth Generation Wireless Networks and Interconnecting Standards; in: *IEEE Personal Communications*, 8 (5), 2001, pp. 8 -9.
- [Jens03] Jenson, S.: Stop Selling \$100 Lemonade!; in: *Proceedings of the Mobile Commerce World Conference*, Melbourne, 2003. <http://www.jensondesign.com/mcw2003.pdf>. Download: 2005-12-29.
- [JoBuTh02] Jones, M., Buchanan, G., Thimbleby, H.: Sorting Out Searching on Small Screen Devices; in: *Proceedings of the 4th International Symposium on Mobile HCI*, Pisa, 2002, pp. 81-94.
- [JoFo88] Johnson, R., Foote, B.: Designing Reusable Classes; in: *Journal of Object-Oriented Programming*, 1 (5), 1988, pp. 22-35.
- [John+02] Johnson, M. et al.: Designing Enterprise Applications with the J2EE Platform, Addison-Wesley, Boston, 2002.

- [**Jone+99**] Jones, M. et al.: Improving Web Interaction on Small Displays; in: *Proceedings of the 8th International Conference on World Wide Web*, Toronto, 1999, pp. 51-59.
- [**JuFu98**] Jul, S., Furnal, G.: Critical Zones in Desert Fog; Aids to Multiscale Navigation; in: *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, San Francisco, 1998, pp. 97-106.
- [**JuMa00**] Jurafsky, D., Martin, J.H: *Speech and Language Processing*, Prentice Hall, London, 2000.
- [**Jura60**] Juran, J.M.: Pareto, Lorenz, Cournot, Bernoulli, Juran and Others, in: *Industrial Quality Control*, 17 (4), 1960, p. 25.
- [**Jura75**] Juran, J.M.: The Non-Pareto Principle: Mea Culpa; in: *Quality Progress*, 8 (5), 1975, pp. 8-9.
- [**Kaas+00**] Kaasinen, E. et al.: Two Approaches to Bringing Internet Services to WAP Devices; in: *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, 2000, pp. 231-246.
- [**Kalj+01**] Kaljuvee, O.: Efficient Web Form Entry on PDAs; in: *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, 2001, pp. 663–672.
- [**Kapp+02**] Kappel, G. et al.: Customizing Web Applications Towards Ubiquity: the Notion and the Issues, Technischer Bericht des Instituts für Anwendungsorientierte Wissensverarbeitung, 2002.
- [**Kari03**] Kariv, M.: Emulating WAP; in: *Web Services Journal*, 3 (6), 2003, pp. 30-33.
- [**KaRo03**] Kaikkonen, A., Roto, V.: Navigating in a Mobile XHTML Application; in: *Proceedings of the Conference on Human Factors in Computing Systems 2003*, Fort Lauderdale, 2003, pp. 329-336.
- [**KaRoSc04**] Karstens, B., Rosenbaum, R., Schuman, H.: Presenting Large and Complex Information Sets on Mobile Handhelds.; in: Deans, P.C. (ed): *E-Commerce and M-Commerce Technologies*, IRM Press, Hershey, London, 2004, pp. 32-56.
- [**Keek97**] Keeker, K.: Improving Website Usability and Appeal, 1997. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsiteplan/html/improvingsiteusa.asp>. Download: 2005-12-12.
- [**KeKi00**] Kerer, C., Kirda, E.: Layout, Content and Logic Separation in Web Engineering; in: *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, 2000, pp. 135-147.
- [**Kend06**] Kendall, P.: Worldwide Cellular User Forecasts, 2005-2010, 2006. <http://www.strategyanalytics.net/default.aspx?mod=ReportAbstractViewer&a0=2739>. Download: 2006-01-30.
- [**Kere+01**] Kerer, C. et al.: Building and Managing XML/XSL-Powered Web Sites: An Experience Report; in: *Proceedings of the 25th International Computer Software and Applications Conference (COMPSAC)*, Chicago, 2001, pp. 547-554.

- [Keto02] Ketola, P.: Integrating Usability with Concurrent Engineering in Mobile Phone development, University of Tampere, A-2002-5, 2002.
- [KiKe00] Kirda, E., Kerer, C: MyXML: An XML Based Template Engine for the Generation of Flexible Web Content; in: *Proceedings of the WEBNET 2000 Conference*, San Antonio, 2000, pp. 317-322.
- [Kilj04] Kiljander. H.: Evolution and Usability of Mobile Phone Interaction Styles, 2004. <http://lib.tkk.fi/Diss/2004/isbn9512273209/isbn9512273209.pdf>. Download: 2005-11-10.
- [KiMa97] Kistler, T., Marais, H.: WebL – A Programming Language for the Web, Digital Equipment Corporation, 1997. <http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-TN-1997-029.pdf>. Download: 2005-10-10.
- [Kird+01] Kirda, E. et al.: Experiences in Engineering Flexible Web Services; in: *IEEE Multimedia*, 8 (1), 2001, pp. 58-65.
- [Kird+03] Kirda, E. et al.: Web Service Engineering with DIWE; in: *Proceedings of the 29th EUROMICRO Conference*, Belek, 2003, pp. 283-290.
- [Klei01] Klein, M.: XML, RDF and Relatives; in: *IEEE Intelligent Systems*, 16 (2), 2001, pp. 26-28.
- [Knig05] Knight, W.: 4G Prototypes Reach Blistering Speeds, 2005-09-02. <http://www.newscientist.com/article.ns?id=dn7943>. Download: 2006-04-13.
- [Knud02] Knudsen, J.: Parsing XML in J2ME, 2002. <http://developers.sun.com/techtopics/mobility/midp/articles/parsingxml/>. Download: 2005-07-09.
- [Knud03] Knudsen, J.: Introduction to Mobile Blogging, 2003. <http://developers.sun.com/techtopics/mobility/midp/articles/blogging/>. Download: 2005-12-12.
- [Knut05] Knutsen, L.: M-Services Expectancies and Attitudes: Linkages and Effects of First Impressions; in: *Proceedings of the 38th Hawaii International Conference on System Sciences*, Big Island, 2005. <http://csdl2.computer.org/comp/proceedings/hicss/2005/2268/03/22680084a.pdf>. Download: 2006-01-02.
- [Koeg03] Keogh, J.: J2ME: The Complete Reference, McGraw-Hill/Osborne, New York, 2003.
- [KoFr02] Kochmer, C., Frandsen, E.: JSP and XML: Integrating XML and Web Services in Your JSP Application, Addison-Wesley, Boston, 2002.
- [KoKoPo01] Kobsa, A., Koenemann, J., Pohl, W.: Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships; in: *Knowledge Engineering Review*, 16 (2), 2001, pp. 111-155.
- [KoSm04] Kowalewski, N., Smith, J.: A Mobile Perspective. Presentation for the W3C Mobile Web Initiative Workshop, Barcelona, 2004. <http://www.w3.org/2004/10/W3C-MWI-Position-Paper-Presentation-2004-11-18.pdf>. Download: 2006-02-03.
- [Krue+01] Krügel, C. et al.: Supporting Multi-Device Enabled Web Services: Challenges and Open Problems; in: *Proceedings of the 10th International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WET ICE)*, Cambridge, 2001, pp. 49-54.

- [**KuDaZa02**] Kurbel, K., Dabkowski, A., Zajac, P.: Software Technology for WAP Based M-Commerce - A Comparative Study of Toolkits for the Development of Mobile Applications; in: *Proceedings of the International Conference WWW/Internet 2002 (IADIS)*, Lisbon, 2002, pp. 673-676.
- [**KuKr06**] Kurbel, K., Krybus, I.: Untersuchung zu praktischem Einsatz und Nutzeffekten des Mobile Business; in: *Proceedings of the MMS Conference*, Passau, 2006, pp. 45-56.
- [**KuTr02**] Kuhlins, S., Tredwell, R.: Toolkits for Generating Wrappers, in: *Proceedings of the NetObject Days 2002 Conference*, Erfurt, 2002, pp. 190-204.
- [**Kwok+04**] Kwok, T. et al.: An Efficient and Systematic Method to Generate XSLT Stylesheets for Different Wireless Pervasive Devices; in: *Proceedings of the 13th International World Wide Web Conference*, New York, 2004, pp. 218-219.
- [**Lehm80**] Lehman M.: Programs, Life Cycles and Laws of Software Evolution; in: *IEEE Special Issue on Software Engineering*, 68 (9), 1980, pp. 1060 -1076.
- [**Lehn02**] Lehner, F.: Mobile Business, Mobile Services, Forschungsbericht der Universität Regensburg, Nr. 49, 2002. <http://www.uni-regensburg.de>. Download: 2005-09-30.
- [**Lehn03**] Lehner, F.: Mobile und drahtlose Informationssysteme. Technologien, Anwendungen, Märkte, Springer-Verlag, Berlin Heidelberg, 2003.
- [**LeKuLe04**] Lehmann, H., Kuhn, J., Lehner, F.: The Future of Mobile Technology: Findings from an European Delphi Study; in: *Proceedings of the 37th Hawaii International Conference on System Sciences*, Hawaii, 2004. <http://csdl2.computer.org/comp/proceedings/hicss/2004/2056/03/205630077b.pdf>. Download: 2005-12-12.
- [**LeLa02**] Lemlouma, T., Layaïda, N.: Content Adaptation and Generation Principles for Heterogeneous Clients; in: *Proceedings of the W3C Workshop on Device Independent Authoring Techniques*, St. Leon-Rot, 2002. <http://opera.inrialpes.fr/people/Tayeb.Lemlouma/Papers/DIAT-PositionPaper.pdf>. Download: 2004-10-10.
- [**LeLa03**] Lemlouma, T., Layaïda, N.: Adapted Content Delivery for Different Contexts; in: *Proceedings of the Symposium on Applications and the Internet*, Orlando, 2003, pp. 190-197.
- [**LeLa04**] Lemlouma, T., Layaïda, N.: Context-Aware Adaptation for Mobile Devices; in: *Proceedings of the IEEE International Conference on Mobile Data Management*, Berkeley, 2004, pp. 106-113.
- [**Levi98**] Levinson, E.: The MIME Multipart/Related Content-Type, RFC 2387, 1998. <http://www.ietf.org/rfc/rfc2387.txt>. Download: 2005-10-10.
- [**Limb+04**] Limbourg, Q. et al.: USIXML: a Language Supporting Multi-Path Development of User Interfaces; in: *Proceedings of the 9th IFIP Working Conference on Engineering for Human-Computer Interaction*, Hamburg, 2004, pp. 89-107.
- [**Ling04**] Ling, R.: The Mobile Connection. The Cell Phone's Impact on Society, Morgan Kaufmann, San Francisco, 2004.

- [Litt04] Little, A. D.: *Mobile Operators: Leaders Hit Back*, Arthur D. Little & Exane Study, 2004.
- [Lu02] Lu, W.W. (Ed.): *Broadband Wireless Mobile: 3G and Beyond*, John Wiley & Sons, West Sussex, 2002.
- [LuCo01] Luyten, K., Coninx, K.: An XML-based Runtime User Interface Description Language for Mobile Computing Devices; in: *Proceedings of the 8th International Workshop on Interactive Systems: Design, Specification and Verification*, Glasgow, 2001, pp. 17-29.
- [Luhn58] Luhn, H.P.: The Automatic Creation of Literature Abstracts; in: *IBM Journal of Research and Development*, 2 (2), 1958, pp. 159-165.
- [LuLa02] Lum, W.Y., Lau, F.C.: A Context-Aware Decision Engine for Content Adaptation; in: *IEEE Pervasive Computing*, 1 (3), 2002, pp. 41-49.
- [Luyt+04] Luyten, K. et al.: Developing User Interfaces with XML: Advances on User Interface Description Languages; in: *Proceedings of the Workshop on Advanced Visual Interfaces*, Gallipoli, 2004.
- [Macr05] Macromedia: Macromedia Flash MX, 2005. <http://www.macromedia.com/software/flash/>. Download: 2005-02-02.
- [Mahm01] Mahmoud, O., H.: *Learning Wireless Java*, O'Reilly & Associates, Beijing, 2001.
- [Mahm04] Mahmoud, Q.: Getting Started With Composite Capabilities/Preference Profile and JSR 188, 2004. <http://developers.sun.com/techtopics/mobility/midp/articles/ccpp/>. Download: 2005-12-12.
- [MaKhRa05] Massey, A., Khatri, V., Ramesh, V.: From the Web to the Wireless Web: Technology Readiness and Usability; in: *Proceedings of the 38th Hawaii International Conference on System Sciences*, Big Island, 2005. <http://csdl2.computer.org/comp/proceedings/hicss/2005/2268/01/22680032b.pdf>. Download: 2005-07-02.
- [MaKi98] Marais, H., Kistler, T.: WebL - a Programming Language for the Web; in: *Proceedings of the 7th International Conference on the World Wide Web*, Brisbane, 1998, pp. 259-270.
- [Mall03] Mallick, M.: *Mobile and Wireless Design Essentials*, John Wiley & Sons, Indianapolis, 2003.
- [Mann05] Mann, K.: *JavaServer Faces in Action*, Manning Publications, Greenwich, 2005.
- [Mara99] Marais, H.: Compaq's Web Language. A Programming Language for the Web, Compaq Systems Research Center (SRC), 1999. <http://www.hpl.hp.com/downloads/crl/web/web1.pdf>. Download: 2005-07-02.
- [MaYaNa01] Masuda, H., Yasutomi, D., Najagawa, H.: How to Transform Tables in HTML for Displaying on Mobile Terminals; in: *Proceedings of the NLPRS2001 Workshop*, Tokyo, 2001. <http://www.afnlp.org/nlprs2001/WS-Paraphrase/pdf/04-masuda.pdf>. Download: 2005-11-10.

- [Mayo02]** Mayora-Ibarra, O.: Generation of Device-Independent User Interfaces; in: *Proceedings of the International Workshop on Research & Development of Human Communication Technologies for Conversational Interaction and Learning*, Puebla, 2002, pp. 1-3.
- [Mazz01]** Mazzocchi, S.: Reducing the Effects of Growth Saturation with the Adoption of a Publishing Framework Based on XML Technologies, 2001. <http://www.betaversion.org/~stefano/papers/thesis.pdf>. Download: 2005-01-04.
- [McGo+03]** McGovern, J. et al.: *Java Web Services Architecture*, Morgan Kaufmann Publishers, San Francisco, 2003.
- [MeFi03]** Meißner, K., Fiala, Z.: Annotating Virtual Web Documents with DynamicsMarks; in: *Proceedings of the Berliner XML Tage '03*, Berlin, 2003, pp. 67-77.
- [Metr05]** M:Metrics.: Survey of US Mobile Subscribers, 2005. <http://www.mmetrics.com/services/syndication.aspx>. Download: 2006-01-01.
- [Metr06]** M:Metrics.: First European Benchmark Survey, 2006. <http://www.mmetrics.com/services/syndication.aspx>. Download: 2006-02-10.
- [Mian99]** Miano, J.: *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*, Addison-Wesley Professional, Boston, 1999.
- [Micr02]** Microsoft Corporation: Microsoft eMbedded Visual Tools, 2002. <http://msdn.microsoft.com/mobility/othertech/eVT/default.aspx>. Download: 2005-09-12.
- [Micr05]** Microsoft Corporation: Component Object Model (COM), 2005. <http://www.microsoft.com/com/default.msp, 2005>. Download: 2005-01-22.
- [MiKaBo04]** Misedakis, I., Kapoulas, V., Bouras, C.: Web Page Fragmentation and Content Manipulation for Constructing Personalized Portals; in: *Proceedings of the 6th Asia Pacific Web Conference (APWeb'04)*, Hangzhou, 2004, pp. 744-754.
- [MoAs02]** Moczar, L.; Aston, J.: *Cocoon Developer's Handbook*, SAMS Publishing, Indianapolis, 2002.
- [Moll05]** Moll, C.: *Mobile Web Design*, 2005. <http://www.cameronmoll.com/archives/000398.html>. Download: 2006-05-12.
- [Mons03]** Monson-Haefel, R.: *J2EE Web Services*, Addison-Wesley Publishing Company, Boston, 2003.
- [MoSc04]** Moura, S., Schwabe, D.: Interface Development for Hypermedia Applications in the Semantic Web; in: *Proceedings of the WebMedia/LA-Web 2004 Conference*, Ribeirão Preto-SP, 2004, pp. 106-113.
- [Muel00]** Müller-Veerse, F.: *Mobile Commerce Report*, Durlacher Corp., London, 2000.
- [Muel04]** Mueller, J.P.: *Mining Google Web Services: Building Applications with the Google API*, Sybex, San Francisco London, 2004.
- [Mull02]** Muller, N.: *Wireless A to Z*, McGraw-Hill Professional, New York, 2002.

- [MuRy96]** Murray, J.D., van Ryper, W.: Encyclopedia of Graphics File Formats, 2nd Edition, O'Reilly and Associates, Sebastopol, 1996.
- [MuScBI04]** Müller, W., Schäfer, R., Bleul, S.: Interactive Multimodal User Interfaces for Mobile Devices; in: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, Big Island, 2004. <http://csdl.computer.org/comp/proceedings/hicss/2004/2056/09/205690286a.pdf>. Download: 2005-11-23.
- [MyBu04]** Myers, B., Burnett, M.: End Users Creating Effective Software; in: *Proceedings of the Conference on Human Factors in Computing Systems*, Vienna, 2004, pp. 1592-1593.
- [MyHuPa00]** Myers, B., Hudson, S., Pausch, R.: Past, Present and Future of User Interface Software Tools; in: *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7 (1), 2000, pp. 3-28.
- [Naga03]** Nagao, K.: Digital Content Annotation and Transcoding, Artech House Publishers, Norwood, 2003.
- [NaShSq01]** Nagao, K., Shirai, Y., Squire, K.: Semantic Annotation and Transcoding: Making Web Content More Accessible; in: *IEEE MultiMedia*, 8 (2), 2001, pp.69-81.
- [NaSiSh05]** Nah, F., Siau, K., Sheng, H.: The Value of Mobile Applications: A Utility Company Study; in: *Communications of the ACM*, 48 (2), 2005, pp. 85-90.
- [NC99]** Netscape Communications: RDF Site Summary (RSS) 0.91 Specification, Revision 3, 1999. <http://my.netscape.com/publish/formats/rss-spec-0.91.html>. Download: 2006-01-02.
- [NewF06]** News Factory: Get Ready for Fourth-Generation Wireless, 2006-09-19. http://www.newsfactor.com/story.xhtml?story_id=11000ATE5566&page=1. Download: 2006-09-26.
- [Niel93]** Nielsen, J.: Usability Engineering, Academic Press, London, 1993.
- [NiRa00]** Nielsen, J., Ramsay, M.: WAP Usability – Déjà Vu: 1994 All Over Again, Norman Nielsen Group Report, 2000.
- [Noki03]** Nokia: The Dawn of Mobile Enterprise: European Intentions to Mobilize Data Applications, Nokia Research Report, 2003. http://www.nokia.com/nfb/pdf/Research_document.pdf. Download: 2005-12-20.
- [Noki04]** Nokia: Mobile Services Market - Status, Evolution and Forecasts, 2004. http://ncsp.forum.nokia.com/download/?asset_id=11991. Download: 2006-01-04.
- [NoSv01]** Novak, L., Svensson, M.: MMS - Building on the Success of SMS; in: *Ericsson Review*, 3/2001, pp. 102-109.
- [NWG05]** Network Working Group: The Atom Syndication Format. Request for Comments: 4287, 2005. <http://www.ietf.org/rfc/rfc4287>. Download: 2006-01-02.
- [OASI03]** OASIS: UDDI Version 3, 2003. <http://www.uddi.org/specification.html>. Download: 2004-06-29.

- [OASI04]** OASIS: The Relationship of the UIML 3.0 Specification to Other Standards/Working Groups, 2004. <http://www.oasis-open.org/committees/download.php/8256/The%20Relationship%20of%20the%20UIML%203%20v01.04.doc>. Download: 2005-12-28.
- [ODro+03]** O'Droma, M. et al. Always Best Connected Enabled 4G Wireless World, in: *Proceedings of the IST Mobile and Wireless Communications Summit*, 2003, pp. 710–716.
- [OMA04]** Open Mobile Alliance: Mobile Web Services Working Group, 2004. http://www.openmobilealliance.org/tech/wg_committees/mws.html. Download: 2005-06-29.
- [OMG04]** Object Management Group: Common Object Request Broker Architecture (CORBA/IIOP) 3.0.3, 2004. http://www.omg.org/technology/documents/corba_spec_catalog.htm. Download: 2005-01-20.
- [Oper06]** OperaOpera Platform: Enabling Web applications on mobile phones, 2006. http://www.opera.com/products/mobile/platform/opera_platform_brochure.pdf. Download: 2006-03-04.
- [Orac05]** Oracle: JDeveloper 10g, 2005. <http://otn.oracle.com/products/jdev/index.html>. Download: 2005-09-20.
- [ORei05]** O'Reilly, T.: What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software, <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>, 2005. Download: 2006-03-04.
- [Orti03]** Ortiz, E.: The MIDP 2.0 Push Registry, 2003. <http://developers.sun.com/techttopics/mobility/midp/articles/pushreg/>. Download: 2005-01-21.
- [Orti04]** Ortiz, E.: Web Services APIs for J2ME, Part 1: Remote Service Invocation API, 2004. <http://www-128.ibm.com/developerworks/wireless/library/wi-jsr>. Download: 2005-10-11.
- [Orti04a]** Ortiz, E.: Web Services APIs for J2ME, Part 2: Java API for XML Processing, 2004. <http://www.ibm.com/developerworks/java/library/wi-xmlparse>. Download: 2005-10-11.
- [OSI05]** Open Source Initiative: The Open Source Definition, 2005. http://www.opensource.org/docs/definition_plain.php. Download: 2005-09-20.
- [PaRePa04]** Papasalouros, A., Retalis, S., Papaspyrou, N.: Semantic Description of Educational Adaptive Hypermedia Based on a Conceptual Model; in: *Educational Technology & Society*, 7 (4), 2004, pp. 129-142.
- [Parn72]** Parnas, D.L.: On the Criteria to Be Used in Decomposing Systems into Modules; in: *Communications of the ACM*, 15 (12), 1972, pp. 1053–1058.
- [Parn76]** Parnas, D.L.: On the Design and Development of Program Families; in: *IEEE Transactions on Software Engineering*, 2 (1), 1976, pp. 1-9.
- [Pass03]** Passani, L.: Welcome to the WURFL; in: *Wireless Business and Technology*, 3 (8), 2003.
- [Patz02]** Patzer, A.: JSP Examples and Best Practices, Apress, Berkeley, 2002.

- [Pede05]** Pedersen, A.: The Mobile Internet: the Pioneering Users Adoption Decisions; in: *Proceedings of the 38th Hawaii International Conference on System Sciences*, Big Island, 2005. <http://csdl2.computer.org/comp/proceedings/hicss/2005/2268/03/22680084b.pdf>. Download: 2006-01-02.
- [PerI05]** Perlin, N.: Device-Independence – Single Sourcing's Other Side; in: *Intercom*, 1/2005, pp. 17-19.
- [Piot05]** Piotrowski, M.: Dangerous Google – Searching for Secrets; in: *Hakin9 Magazine*, 4/2005, pp. 1-12.
- [Piro02]** Piroumian, V.: *Wireless J2ME Platform Programming*, Prentice Hall, London, 2002.
- [Pous+04]** Pousttchi, K. et al.: Mobile Enterprise in Germany: State-of-the-Art, Expectations and Perspectives for Mobile Business Processes in Small and Medium-Sized Enterprises on the German Market, Joint Study by the Finnish Foreign Trade Association (FINPRO) and the University of Augsburg, Chair of Business Informatics and Systems Engineering, Munich, 2004.
- [Powe03]** Powers, S.: *Practical RDF*, O'Reilly & Associates, Beijing, 2003.
- [Pras99]** Prasad, N.: GSM Evolution Towards Third Generation UMTS/IMT2000; in: *Proceedings of the IEEE International Conference on Personal Wireless Communications*, Jaipur, 1999, pp. 50-54.
- [PrDiWo02]** Prohm, B., Dittner, P., Wood, B.: *Mobile Terminals Statistics Worldwide, 1996-2005*. Gartner Dataquest, TCM-WW-MS-0174, 2002.
- [Pree+94]** Preece, J. et al.: *Human Computer Interaction*, Addison-Wesley, Wokingham, 1994.
- [PrRu03]** Prasad, R., Ruggieri, M.: *Technology Trends in Wireless Communications*, Artech House Publishers, London, 2003.
- [PuEi01]** Puerta, A., Eisenstein, J.: *XIML: A Universal Language for User Interfaces*, 2001. <http://www.xml.org/documents/XimlWhitePaper.pdf>. Download: 2005-09-26.
- [PuEi02]** Puerta, A., Eisenstein, J.: *XIML: A Common Representation for Interaction Data*; in: *Proceedings of the ACM Conference on Intelligent User Interfaces (IUI '02)*, San Francisco, 2002, pp. 214-215.
- [Puli03]** Pulier, E.: The Reality, Challenges, and Enormous Potential of Web Services, in: *Web Services Journal*, 5/2003, pp. 48-50.
- [Qual02]** Qualcomm: *Creating a BREW Application from Scratch*, 2002. <http://www.qualcomm.com/about/whitepaper.pdf>. Download: 2005-09-12.
- [RaAI02]** Rahman, A., Alam, H.: Challenges in Web Document Summarization: Some Myths and Reality; in: *Proceedings of the 9th Document Recognition and Retrieval Conference*, Santa Clara, 2002.
- [RaAIHa01]** Rahman, A., Alam, H., Hartono, R.: Content Extraction from HTML Documents; in: *Proceedings of the Web Document Analysis Workshop*, Seattle, 2001, pp. 7-10.

- [RAB02] RSS Advisory Board: Really Simple Syndication (RSS) 2.0 Specification, 2002. <http://www.rssboard.org/rss-specification>. Download: 2006-01-02.
- [RaJeSc01] Rauschenbach, U., Jeschke, S., Schumann, H.: General Rectangular FishEye Views for 2D Graphics; in: *Computers and Graphics*, 25 (4), 2001, pp. 609-617.
- [RaScFi05] Rannenberg, K., Schneider, I., Figge, S.: Mobile Systeme und Anwendungen - Hammer sucht Nagel; in: *Wirtschaftsinformatik*, 47 (1), 2005, pp. 1-2.
- [Rasm+04] Rasmusson, J. et al.: Multimedia in Mobile Phones - The Ongoing Revolution; in: *Ericsson Review*, 2/2004, pp. 98-107.
- [Real00] RealNetworks: RealSystem Production Guide, 2000. <http://service.real.com/help/library/guides/production8/htmfiles/smilext.htm>. Download: 2005-12-02.
- [Redm97] Redmond, F.: DCOM: Microsoft Distributed Component Object Model, John Wiley & Sons, New York, 1997.
- [ReMa03] Read, K., Maurer, F.: Developing Mobile Wireless Applications; in: *IEEE Internet Computing*, 7 (1), 2003, pp. 81-86.
- [Rock01] Rockwell, W.: XML, XSLT, Java., and JSP: A Case Study in Developing a Web Application, New Riders, Indianapolis, 2001
- [Roge95] Rogers, E.M.: Diffusion of Innovations, Free Press, New York, 1995.
- [Roma+05] Roman, E. et al.: Mastering Enterprise JavaBeans, 3rd Edition, Wiley Publishing, Indianapolis, 2005.
- [RoPi04] Roberts, G., Pick, J.: Technology Factors in Corporate Adoption of Mobile Cell Phones: A Case Study Analysis; in: *Proceedings of the 37th Hawaii International Conference on System Sciences*, Big Island, 2004. <http://csdl.computer.org/comp/proceedings/hicss/2004/2056/09/205690287b.pdf>. Download: 2005-12-12.
- [RWG00] RSS-DEV Working Group: RDF Site Summary (RSS) 1.0 Specification, 2000. <http://web.resource.org/rss/1.0/>. Download: 2006-01-02.
- [Rysa04] Rysavy, P.: Brush Up on Bluetooth; in: *Network Computing*, 2004, pp. 86-88. <http://www.rysavey.com/Articles/Bluetooth.pdf>. Download: 2005-01-02.
- [Rysa04a] Rysavy, P.: Forever Evolving; in: *Network Computing*, 2004, pp. 40-49. <http://www.rysavey.com/Articles/ForeverEvolving.pdf>. Download: 2005-01-02.
- [RysR05] Rysavy Research: Data Capabilities: GPRS to HSDPA and Beyond, White Paper, 2005. http://www.rysavey.com/Articles/Rysavy_Data_Paper-Sept2005.pdf. Download: 2005-12-12.
- [SaHj01] Saryanarayana, L., Hjelm, J.: CC/PP for Context Negotiation and Contextualization; in: *Proceedings of the 2nd International Conference on Mobile Data Management (MDM)*, Hong Kong, pp. 239-245.
- [SALF02] SALT Forum: Speech Application Language Tags (SALT), 1.0 Specification, 2002. <http://www.saltforum.org/saltforum/downloads/SALT1.0.pdf>. Download: 2005-01-03.

- [Scan04]** Scanlon, L.: Rethinking the Computer - Project Oxygen is Turning out Prototype Computer Systems; in: *Technology Review*, July/August 2004. <http://www.technologyreview.com/articles/scanlon0704.asp?p=1>. Download: 2005-12-02.
- [Sche00]** Scheemaecker, M.: NanoXML Parser. <http://nanoxml.sourceforge.net>. Download: 2005-10-10.
- [Schi+01]** Schilit, B.N., Trevor, J., Hilbert, D., Koh, T.: m-Links: an Infrastructure for Very Small Internet Devices; in: *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001, Rome, pp. 122-131.
- [Schi+02]** Schilit, B.N. et al.: Web Interaction Using Very Small Internet Devices; in: *Computer*, 35 (10), 2002, pp. 37-45.
- [Schm05]** Schmidt, M.: List of Java Libraries to Read and Write Image Files, 2005. <http://www.geocities.com/marcoschmidt.geo/java-image-coding.html>. Download: 2005-12-01.
- [Schw+04]** Schwabe, D. et al.: Design and Implementation of Semantic Web Applications; in: *Proceedings of the Workshop on Application Design, Development and Implementation Issues in the Semantic Web (WWW 2004)*, Manhattan, 2004.
- [ScKo03]** Scharf, D.; Koch, A.: A New Technology-Driven Approach for Extended Mobile Applications: MAML - Mobile Application Markup Language; in: *Proceedings of the 8th International Workshop on Mobile Multimedia Communications (MoMuC)*, Munich, 2003, pp. 23-27.
- [SeDoKI05]** Seffah, A., Donyaee, M., Kline, R.: Usability and Quality in Use Measurement and Metrics: An Integrative Model, 2005. <http://hci.cs.concordia.ca/www/hcse/projects/humancase/QUIM.pdf>. Download: 2005-09-12.
- [Seki02]** Seki, Y.: Sentence Extraction by TF/IDF and Position Weighting from Newspaper Articles; in: *Proceedings of the 3rd NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, Tokyo, 2002, pp. 55-59.
- [Sepp04]** Seppänen, J.: Mobile Audio Formats, 2004. <http://www.iasig.org/members/mawg/MobileAudioFormats2004-10-26.pdf>. Download: 2005-02-02.
- [SeSaPi03]** Seurre, E., Savelli, P., Pietri, J.P.: GPRS for Mobile Internet, Artech House, Boston, 2003.
- [ShChRy01]** Shachor, G., Chace, A., Rydin, M.: JSP Tag Libraries, Manning Publications, Greenwich, 2001.
- [ShAdWa94]** Shilit, B., Adams, N., Want, R.: Context-Aware Computing Applications, in: *Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications*, Santa Cruz, 1994, pp. 85-90.
- [ShTr01]** Shalloway A., Trott J.: Design Patterns Explained: A New Perspective on Object-Oriented Design, Addison-Wesley, Boston, 2001.

- [Sigu01] Sigurdson, J.: WAP: OFF – Origin, Failure, Future, Japanese-European Technology Studies (JETS), 2001. <http://www.telecomvisions.com/articles/pdf/wap-off.pdf>. Download: 2005-10-23.
- [Simo+04] Simon, R. et al.: A Generic UIML Vocabulary for Device- and Modality Independent User Interfaces; in: *Proceedings of the 13th International Conference on World Wide Web*, New York, 2004, pp. 434-435.
- [SkJoDi00] Skiba, B., Johnson, M., Dillon, M.: *Moving in Mobile Media Mode*, Lehman Brothers, 2000.
- [SmBrKr01] Smiley, K., Brownlee, T., Kriebel, N.: *IT Trends: Managing Mobility*, Giga Information Group, 2001.
- [SoVa03] Souchon, N., Vanderdonckt, J.: A Review of XML-compliant User Interface Description Languages; in: *Proceedings of the 10th International Conference on Design, Specification, and Verification of Interactive Systems*, Madeira, 2003, pp. 377-391.
- [SoVaBo01] Souchon, N., Vanderdonckt, J., Bouillon, L.: Flexible Reverse Engineering of Web Pages with VAQUITA; in: *Proceedings of the 8th Working Conference on Reverse Engineering*, Stuttgart, 2001, pp. 241-248.
- [SPGM05] SPG Media: NTT DoCoMo 4G Network and I-mode Technology, 2005. http://www.mobilecomms-technology.com/projects/4g_i-mode/. Download: 2006-01-01.
- [SpGo02] Springer, T., Göbel, S.: A Modular Adaptation Framework for Single Source Publishing; in: *Proceedings of the IADIS International Conference WWW/Internet 2002*, Lisbon, 2002, pp. 11-19.
- [Spri+03] Priestersbach, A. et al.: A Single Source Authoring Language to Enhance the Access from Mobile Devices to Web Enterprise Applications; in: *Proceedings of the 12th International World Wide Web Conference*, Budapest, 2003.
- [Srin95] Srinivasan, R.: RPC: Remote Procedure Call Protocol Specification Version 2, RFC 1831, 1995. <http://www.ietf.org/rfc/rfc1831.txt>. Download: 2005-10-10.
- [Sriv04] Srivastava, L.: Shaping the Future Mobile Information Society: The Case of Japan; in: *Proceedings of the ITU/MIC Workshop on Shaping the Future Mobile Information Society*, Seoul, 2004. <http://www.itu.int/osg/spu/ni/futuremobile/general/casestudies/JapancaseLS.pdf>. Download: 2005-09-02.
- [StMu02] Strahan, R., Muldoon, C.: State of the Art Mobile Computing Devices; University College Dublin Deliverable e=mc².1.1.2.2002, 2002. <http://emc2.ucd.ie/deliverables/e=mc2.1.1.2.2002.doc>. Download: 2005-10-12.
- [StPa02] Steinberg, J., Pasquale, J.: A Web Middleware Architecture for Dynamic Customization of Content for Wireless Clients; in: *Proceedings of the 11th World Wide Web Conference*, Honolulu, 2002, pp. 639-650.
- [StRu98] Stirewalt, K.; Rugaber, S.: Automatic User-Interface Generation by Model Composition; in: *Proceedings of the 13th IEEE International Conference on Automated Software Engineering*, Honolulu, 1998, pp. 177-187.

- [StTe05]** Steinert, M., Teufel, S.: The European Mobile Data Service Dilemma; in: *Proceedings of the IFIP International Working Conference on Mobile Information Systems*, Leeds, 2005, pp. 63-78.
- [StZh00]** Stasko, J., Zhang, E.: Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filtering Hierarchy Visualizations; in: *Proceedings of the IEEE Conference on Information Visualization*, Salt Lake City, 2000, pp. 57-65.
- [Sun03]** Sun Microsystems: Java Specification Request 188: CC/PP Processing, 2003. <http://jcp.org/en/jsr/detail?id=188>. Download: 2005-02-01.
- [Sun03a]** Sun Microsystems: Java Specification Request 52: A Standard Tag Library for JavaServer Pages, 2003. <http://jcp.org/en/jsr/detail?id=52>. Download: 2005-02-01.
- [Sun04]** Sun Microsystems: Java Specification Request 154: Java Servlet 2.4 Specification, 2004. <http://jcp.org/en/jsr/detail?id=154>. Download: 2005-02-01.
- [Sun04a]** Sun Microsystems: Java Specification Request 172: J2ME Web Services Specification, 2004. <http://jcp.org/en/jsr/detail?id=172>, 2004. Download: 2005-03-21.
- [Sun04b]** Sun Microsystems: Java Specification Request 220: Enterprise JavaBeans 3.0, 2004. <http://jcp.org/en/jsr/detail?id=220>. Download: 2005-01-12.
- [Sun04c]** Sun Microsystems: Java Specification Request 252: JavaServer Faces 1.2, 2004. <http://jcp.org/en/jsr/detail?id=252>. Download: 2005-01-12.
- [Sun04d]** Sun: JSR188, J2EE CC/PP Processing Technology. <http://java.sun.com/j2ee/ccpp>, 2004. Download: 2005-12-12.
- [Sun05]** Sun Microsystems: J2ME Web Services APIs (WSA), 2005. <http://java.sun.com/products/wsa>. Download: 2005-10-11.
- [Sun05a]** Sun Microsystems: Java Specification Request 245: JavaServer Pages 2.1, 2005. <http://jcp.org/aboutJava/communityprocess/edr/jsr245/index.html>. Download: 2005-02-01.
- [Sun05b]** Sun Microsystems: Java Specification Request 267: JSP Tag Library for Web Services, 2005. <http://jcp.org/en/jsr/detail?id=267>. Download: 2005-02-01.
- [Sun06]** Sun Microsystems: Java API for XML Messaging (JAXM), 2006. <http://java.sun.com/webservices/jaxm/index.jsp>. Download: 2006-06-02.
- [Sun06a]** Sun Microsystems: Java API for XML Processing (JAXP), 2006. <http://java.sun.com/webservices/jaxp/index.jsp>. Download: 2006-04-03.
- [Sun06b]** Sun Microsystems: Java API for XML Registries (JAXR), 2006. <http://java.sun.com/webservices/jaxr/index.jsp>. Download: 2006-03-01.
- [Sun06c]** Sun Microsystems: Java API for XML Web Services (JAX-WS), 2006. <http://java.sun.com/webservices/jaxws/index.jsp>. Download: 2006-05-02.
- [Sun06d]** Sun Microsystems: Java API for XML-Based RPC (JAX-RPC), 2006. <http://java.sun.com/webservices/jaxrpc/index.jsp>. Download: 2006-03-03.

- [Sun06e] Sun Microsystems: Java Architecture for XML Binding (JAXB), 2006. <http://java.sun.com/webservices/jaxb/index.jsp>. Download: 2006-07-01.
- [Sun06f] Sun Microsystems: SOAP with Attachments API for Java (SAAJ), 2006. <http://java.sun.com/webservices/saaj/index.jsp>. Download: 2006-06-02.
- [Sun06g] Sun Microsystems: The Java Web Services Tutorial, 2006. <http://java.sun.com/webservices/docs/1.6/tutorial/doc/JavaWSTutorial.pdf>. Download: 2006-06-20.
- [Suro04] Surowiecki, J.: *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, Doubleday, New York, 2004.
- [Sutt02] Sutton, A.: ASXMLP Parser, 2002. <http://sourceforge.net/projects/asxmlp/>. Download: 2005-10-10.
- [Szek+95] Szekely, P. et al.: Declarative Interface Models for User Interface Construction Tools: the MASTERMIND Approach; in: *Proceedings of the 7th Working Conference on Engineering for Human-Computer Interaction*, Yellowstone Park, 1995, pp. 120-150.
- [Szek96] Szekely P.: Retrospective and Challenges for Model-Based Interface Development; in: *Proceedings of the 3rd International Eurographics Workshop on Design, Specification and Verification of Interactive Systems*, Namur, 1996, pp. 1-27.
- [Szyp98] Szyperski, C.: *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, Boston, 1998.
- [Tara02] Tarasewich P.: Issues in Mobile E-Commerce; in: *Communications of the Association for Information Systems*, 8/2002, pp. 41-64.
- [Tara02a] Tarasewich, P.: Wireless Devices for Mobile Commerce: User Interface Design and Usability; in: Mennecke, B., Strader, T. (Eds.): *Mobile Commerce: Technology, Theory, and Applications*, Idea Group Publishing, Hershey, 2002, pp. 26-50.
- [TaSt02] Tanenbaum A., S, Steen, S.: *Distributed Systems: Principles and Paradigms*, Prentice Hall, London, 2002.
- [TeC03] TechConsult GmbH: *Mobile Business in Deutschland*, Kassel, 2003. <http://www.techconsult.de/Studien/docs/BERICHTMobileBusinessinDeutschland.pdf>. Download: 2005-01-02.
- [Tee03] Tee, R.: Contextualizing the Mobile Internet - A SCOT Approach to Mobile Internet Services in Europe and Japan, EC/DC Infonomics Report, 2003. http://ecdc.info/publications/reports/2003_05_rt_mobile.pdf. Download: 2005-12-23.
- [Tee05] Tee, R.: Contextualizing the Mobile Internet; in: Lasen, A., Hamill, L. (Eds.): *Wireless World: Mobiles - Past, Present and Future*, Springer Verlag, London, 2005.
- [Thau+05] Thauvin, E. et al.: Google Tag Library, 2005. <http://google-taglib.sourceforge.net/index.html>. Download: 2006-04-06.

- [Thom00] Thomason, L.: TinyXML Parser, 2000. <http://www.grinninglizard.com/tinyxml/>. Download: 2005-10-10.
- [Topl02] Topley, K.: J2ME in a Nutshell, O'Reilly & Associates, Sebastopol, 2002.
- [Topl03] Topley, K.: Java Web Services in a Nutshell, O'Reilly, Beijing, 2003.
- [Tras05] Trasattii, A.: WURFL Capabilities Part 1: Overview of the Browsing Related Capabilities, 2005. http://developer.openwave.com/dvl/tools_and_sdk/wurfl_and_wall/wurfl_capabilities_part_1.htm. Download: 2005-10-10.
- [Trev+01] Trevor, J. et al.: Form Desktop to Phonetop: A UI for Web Interaction on Very Small Devices; in: *Proceedings of the 14th Annual ACM Symposium on User Interfaces Software and Technology*, Orlando, 2001, pp. 121-130.
- [Tsch+02] Tscheligi, M. et al.: Empirical Usability Studies on User Interface Modules and Elements: a Prerequisite of Usable Applications Specifically Tailored to Different Mobile Devices; in: *Proceedings of the 7th Meeting of the Wireless World Research Forum*, Eindhoven, 2002.
- [Tsuc00] Tsuchiyama, R.: Deconstructing "Phone Culture". How Japan Became a Leader in Mobile Internet; in: *Journal of the American Chamber of Commerce in Japan*, 2000. <http://www.dnso.org/clubpublic/registrars/Arc01/msg00060.html>. Download: 2005-10-23.
- [Turn03] Turner, J.: Chiba Cookbook, 2003. <http://chiba.sourceforge.net/ChibaCookBook.pdf>. Download: 2005-09-22.
- [UMTS03] UMTS Forum: Mobile Evolution: Shaping the Future, 2003. http://www.umts-forum.org/servlet/dycon/ztumts/umts/Live/en/umts/MultiMedia_PDFs_Papers_Paper-1-August-2003.pdf. Download: 2005-10-31.
- [UnP99] Unwired Planet: HDML Language Reference 3.0, 1999. <http://developer.openwave.com/html/doc/31h/hdmlref/output/front.html>. Download: 2005-09-21.
- [VaFl04] Vanderdonckt, J., Florins, M.: Graceful Degradation of User Interfaces as a Design Method for Multiplatform Systems; in: *Proceedings of the 9th International Conference on Intelligent User Interface*, Funchal, 2004, pp. 140-147.
- [VaJa01] Varshney, U., Jain, R.: Issues in Emerging 4G Wireless Networks; in: *IEEE Computer*, 34 (6), 2001, pp. 94-96.
- [Vand+00] Vanderdonckt, J. et al.: USIXML: a User Interface Description Language for Specifying Multimodal User Interfaces; in: *Proceedings of the W3C Workshop on Multimodal Interaction (WMI2004)*, Sophia Antipolis, 2004. <http://www.w3.org/2004/02/mmi-workshop/vanderdonckt-louvain.pdf>. Download: 2005-01-23.
- [Vand+04] Vanderdonckt, J. et al.: USIXML: a Language Supporting Multi-Path Development of User Interfaces; in: *Proceedings of the 9th IFIP Working Conference on Engineering for Human-Computer Interaction*, Hamburg, 2004. <http://www.usixml.org/index.php?download=EHCI-DSVIS04-usiXML.pdf>. Download: 2005-10-23.

- [Vars+02] Varshney, U. et al.: *Wireless in the Enterprise: Requirements and Possible Solutions*; in: *Proceedings of the Workshop on Wireless Strategy in the Enterprise: An International Research Perspective*, Berkeley, 2002.
- [Venk+03] Venkatesh, V. et al.: *User Acceptance of Information Technology: Towards a Unified View*; in: *MIS Quarterly*, 27 (3), 2003, pp. 425-478.
- [VeRaMa03] Venkatesh, V., Ramesh, V., Massey, A.P.: "e" ≠ "m": *Ramifications for Wireless Design*; in: *Communications of the ACM*, 46 (12), 2003, pp. 53-56.
- [Vino02] Vinoski, S.: *Putting the "Web" into Web Services*; in: *IEEE Internet Computer*, 6 (4), 2002, p. 90-92.
- [Vino02a] Vinoski, S.: *Web Services Interaction Models: Current Practice*; in: *IEEE Internet Computer*, 6 (3), 2002, p. 89-91.
- [Vlis02] Vlist van der, E.: *XML Schema: The W3C's Object-Oriented Descriptions for XML*, O'Reilly, Beijing, 2002.
- [Voge03] Vogels, W.: *Web Services Are Not Distributed Objects*; in: *IEEE Internet Computing*, November/December 2003, pp. 59-66.
- [W3C00] W3C: *SOAP Messages with Attachments*, W3C Note, 2000. <http://www.w3.org/TR/SOAP-attachments>. Download: 2005-10-04.
- [W3C00a] W3C: *XHTML Basic Recommendation*, 2000. <http://www.w3.org/TR/xhtml1-basic/>. Download: 2005-01-04.
- [W3C01] W3C: *Modularization of XHTML*, 2001. <http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/Overview.html#toc>. Download: 2005-01-04.
- [W3C01a] W3C: *WSDL 1.1*. <http://www.w3.org/TR/wsdl>, 2001. Download: 2005-06-24.
- [W3C01b] W3C: *XHTML+Voice Profile 1.0*, 2001. <http://www.w3.org/TR/xhtml+voice>. Download: 2005-9-01.
- [W3C01c] W3C: *XML Schema, Part 0 - Primer*, 2001. <http://www.w3.org/TR/xmlschema-0/>. Download: 2005-09-23.
- [W3C02] W3C: *Authoring Scenarios for Device Independence*, 2002. <http://www.w3.org/2001/di/public/as>. Download: 2005-09-13.
- [W3C02a] W3C: *CSS Mobile Profile 1.0*, 2002. <http://www.w3.org/TR/css-mobile>. Download: 2005-10-07.
- [W3C02b] W3C: *Delivery Context Overview for Device Independence*, 2005. <http://www.w3.org/TR/di-dco/>. Download: 2005-02-21.
- [W3C02c] W3C: *Device Independence Working Group Charter*, 2002. <http://www.w3.org/2002/06/w3c-di-wg-charter-20020612.html>. Download: 2005-12-22.
- [W3C02d] W3C: *Media Queries W3C Candidate Recommendation*, 2002. <http://www.w3.org/TR/css3-mediaqueries/>. Download: 2005-09-10.

- [W3C02e]** W3C: XHTML 1.0, the Extensible HyperText Markup Language Specification, 2002. <http://www.w3.org/TR/xhtml1/>. Download: 2005-09-21.
- [W3C02f]** W3C: XML Pointer Language (XPointer), 2002. <http://www.w3.org/TR/xptr/>. Download: 2005-10-20.
- [W3C03]** W3C: Authoring Challenges for Device Independence, 2003. <http://www.w3.org/TR/acdi/>. Download: 2005-08-23.
- [W3C03a]** W3C: Device Independence Principles, 2003. <http://www.w3.org/TR/di-princ/>. Download: 2005-09-21.
- [W3C03b]** W3C: Mobile SVG Profiles: SVG Tiny and SVG Basic, 2003. <http://www.w3.org/TR/SVGMobile/>. Download: 2005-09-02.
- [W3C03c]** W3C: Scalable Vector Graphics (SVG) 1.1 Specification, 2003. <http://www.w3.org/TR/SVG11/>. Download: 2005-09-21.
- [W3C03d]** W3C: SOAP 1.2, 2003. <http://www.w3.org/TR/SOAP/>. Download: 2005-06-29.
- [W3C03e]** W3C: XSL Transformations (XSLT) Version 2.0, 2003. <http://www.w3.org/TR/xslt20/>. Download: 2005-08-23.
- [W3C04]** W3C: Authoring Techniques for Device Independence, 2004. <http://www.w3.org/TR/di-atdi/>. Download: 2005-10-03.
- [W3C04a]** W3C: Composite Capabilities/Preference Profile (CC/PP) Structure and Vocabularies 1.0, 2004. <http://www.w3.org/TR/CCPP-struct-vocab/>. Download: 2005-09-12.
- [W3C04b]** W3C: Extensible Markup Language (XML) 1.0 (3rd Edition), 2004. <http://www.w3.org/TR/REC-xml/>. Download: 2005-06-23.
- [W3C04c]** W3C: OWL Web Ontology Language Overview, W3C Recommendation, 2004. <http://www.w3.org/TR/owl-features/>. Download: 2005-01-02.
- [W3C04d]** W3C: RDF Primer, 2004. <http://www.w3.org/TR/rdf-primer/>. Download: 2005-06-27.
- [W3C04e]** W3C: RDF Schema Specification 1.0, 2004. <http://www.w3.org/TR/rdf-schema/>. Download: 2005-07-03.
- [W3C04f]** W3C: Voice Extensible Markup Language (VoiceXML) Version 2.0, 2004. <http://www.w3.org/TR/voicexml20/>. Download: 2005-10-20.
- [W3C04g]** W3C: Web Services Architecture Requirements, 2004. <http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211/>. Download: 2005-06-29.
- [W3C04h]** W3C: Web Services Architecture, W3C Working Group Note, 2004. <http://www.w3.org/TR/ws-arch/>. Download: 2005-02-10.
- [W3C04i]** W3C: XForms 1.1, 2004. <http://www.w3.org/TR/xforms-11-req/>. Download: 2005-07-06.
- [W3C04j]** W3C: XML Path Language (XPath) 2.0, 2004. <http://www.w3.org/TR/xpath20/>. Download: 2005-10-20.

- [W3C05] W3C: Mobile Web Best Practice Working Group Charter, 2005. <http://www.w3.org/2005/01/BPWGCharter/Overwiev.html>. Download: 2005-12-22.
- [W3C05a] W3C: Mobile Web Best Practices 1.0, 2005. <http://www.w3.org/TR/2005/WD-mobile-bp-20051017/>. Download: 2005-12-22.
- [W3C05b] W3C: Synchronized Multimedia Integration Language (SMIL 2.0), 2nd Edition, 2005. <http://www.w3.org/TR/SMIL/>. Download: 2005-02-02.
- [W3C06] W3C: Cascading Style Sheets Level 2 (CSS 2.1), 2006. <http://www.w3.org/TR/CSS21>. Download: 2006-04-12.
- [W3C06a] W3C: Mobile Web Best Practices 1.0, W3C Candidate Recommendation. <http://www.w3.org/TR/mobile-bp/>. Download: 2006-07-01.
- [W3C97] W3C: Proposal for a Handheld Device Markup Language, 1997. <http://www.w3.org/pub/WWW/TR/NOTE-Submission-HDML.html>. Download: 2005-09-21.
- [W3C98] W3C: Compact HTML for Small Information Appliances, 1998. <http://www.w3.org/TR/1998/NOTE-compactHTML-19980209>. Download: 2005-09-21.
- [W3C99] W3C: HTML 4.0 Guidelines for Mobile Access, 1999. <http://www.w3.org/TR/NOTE-html40-mobile/>. Download: 2005-06-23.
- [W3C99a] W3C: HTML 4.01 Specification, 1999. <http://www.w3.org/TR/html4/>. Download: 2005-09-21.
- [W3C99b] W3C: WAP Binary XML Content Format, 1999. <http://www.w3.org/TR/wbxml>. Download: 2005-08-21.
- [Wall+02] Wallace, P. et al.: i-mode Developer's Guide, Addison-Wesley, Boston, 2002.
- [WAPF01] WAP Forum: WAG UAProf, 2001. <http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf>. Download: 2005-12-12.
- [WAPF01a] WAP Forum: XHTML Mobile Profile, 2001. <http://www.wapforum.org/tech/documents/WAP-277-XHTMLMP-20011029-a.pdf>. Download: 2005-01-04.
- [WAPF02] WAP Forum: Wireless Application Protocol WAP 2.0, Technical White Paper, 2002. http://www.wapforum.org/what/WAPWhite_Paper1.pdf. Download: 2005-08-28.
- [WAPF02a] WAP Forum, WAP CSS Specification Version 1.0, 2002. <http://www.wapforum.org/tech/documents/WAP-239-WCSS-20011026-a.pdf>. Download: 2005-08-10.
- [WaSeAI03] Walke, B., Seidenberg, P., Althoff, M.P.: UMTS: The Fundamentals, John Willey and Sons, West Sussex, 2003.
- [WeGr02] Van Welie, M., de Groot, B.: Consistent Multi-Device Design Using Device Categories; in: *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, Pisa, 2002, pp. 315-318.
- [Wegs+04] Wegscheider, F. et al.: A Multimodal Interaction Manager for Device Independent Mobile Applications; in: *Proceedings of the 13th International Conference on World Wide Web*, New York, 2004, pp. 272-273.

- [Weis02]** Weiss, S.: *Handheld Usability*, John Wiley & Sons, Chichester, 2002.
- [WeMe05]** Web Methods: Making the Transformation to Service-Oriented Architecture, 2005. http://www1.webmethods.com/PDF/Making_the_Transformation_to_SOA.pdf#search=%22A%20March%202005%20survey%20of%20100%20CIOs%20%2B%20Smith%20Barney%20%2B%20SOA%22. Download: 2006-01-02.
- [WiBa04]** Wigdor, D., Balakrishnan, R.: A Comparison of Consecutive and Concurrent Input Text Entry Techniques for Mobile Phones; in: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vienna, 2004, pp. 81-88.
- [WiFo05]** WiMax Forum: WiMax Technology, 2005. <http://www.wimaxforum.org/technology>. Download: 2005-09-01.
- [Wilk04]** Wilkinson, J.: Stylesheets for Handheld Devices, 2004. <http://css-discuss.incutio.com/?page=HandheldStylesheets>. Download: 2005-12-12.
- [WiLo02]** Wilde, E., Lowe, D.: *XPath, XLink, XPointer, and XML: A Practical Guide to Web Hyperlinking and Transclusion*, Addison-Wesley, Boston, 2002.
- [WiMa04]** Wilson, M., Matthews, B.: The Future of the World Wide Web; in: *Proceedings of the 21st Annual British National Conference on Databases*, Edinburgh, 2004, pp. 4-15.
- [Wind04]** Windrum, P.: Leveraging Technological Externalities in Complex Technologies: Microsoft's Exploitation of Standards in the Browser Wars; in: *Research Policy*, 33/2004, pp. 385-394.
- [Wine05]** Winer, D.: OPML 2.0 Draft Specification, 2005. <http://www.opml.org>. Download: 2006-01-02.
- [Wine99]** Winer, D.: XML-RPC Specification, 1999. <http://www.xmlrpc.com/spec>. Download: 2005-02-02.
- [WiPa00]** Wilson Partnership: MINML Parser, 2000. <http://www.wilson.co.uk/xml/minml.htm>. Download: 2005-10-10.
- [ZhHeMi04]** Zhang, Y., Heywood, N., Milios, E.: World Wide Web Site Summarization; in: *Web Intelligence and Agent Systems*, 2 (1), 2004, pp. 39-53.
- [Zieg+04]** Ziegert, T. et al.: Practical Experiences with Device Independent Authoring Concepts; in: *Proceedings of the Workshop on Advanced Visual Interfaces*, Gallipoli, 2004, pp. 17-24.
- [ZiVaGi02]** Zimmermann, G., Vanderheiden, G., Gilman, A.: Universal Remote Console Prototyping of an Emerging XML Based Alternate User Interface Access Standard; in: *Proceedings of the 11th International World Wide Web Conference*, Honolulu, 2002. <http://www2002.org/CDROM/poster/163/>. Download: 2005-10-10.
- [Zobe01]** Zobel, J.: *Mobile Business und M-Commerce: Die Märkte der Zukunft erobern*, Carl Hanser Verlag, München, 2001.

Appendix 1: Examples of phones from various classes of mobile devices

Nokia 6100 (3100)

Category: Web-enabled phones

Web-enabled phones are generally old-fashioned phones supporting WML, with a small screen and designated mainly for conversations and not for browsing.

- | | |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Display | - High-resolution, illuminated, full-graphics display
- Up to 8 lines for text, numbers, and graphics |
| Tri-band operation | - EGSM 900, GSM 1800 and GSM 1900 networks in Europe, Africa, Asia, North and South America; automatic switching between bands |
| Data transfer | - GPRS (General Packet Radio Service)
- HSCSD (High Speed Circuit Switched Data) |
| Messaging | - Text messaging: concatenated SMS, send and receive up to 3 messages
- Picture messaging: Send pictures with text to other compatible phones
- Multimedia messaging (text, audio files, images) |
| Browsing | - Browser supporting WML (WAP 1.2.1) |



Samsung SGH-i300

Category: Low-end smart phones

Low-end smartphones are mobile phones with a number of additional features outside the usual wireless devices such as MP3, camera, MMS, games, Java ME or e-mail. Examples might include Orange's SPV C500 or Motorola A1000.

- | | |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Display | - High-resolution color display
- Supports 262 144 colours colors, 240x320pixels
- 10 lines for text in basic mode with 16 pixel font |
| Tri-band operation | - EGSM 900, GSM 1800, and GSM 1900 networks, automatic switching between bands |
| Data transfer | - GPRS (General Packet Radio Service) |
| Java | - Support for download of games and applications (J2ME) MIDP 2.0 |
| Messaging | - Multimedia messaging: send and receive messages containing text, images, audio, and video.
- Email: supports POP3, SMTP, IMAP4, APOP, SSL, TLS protocols
- Instant Messaging (Pocket MSN) |
| Browsing | - Browser supporting WML (WAP 2.0) |
| Memory | - 64 MB of memory, 3 GB hard drive |
| Audio/visual media features | - Camera: 1.3 Megapixel resolution (1280x1024), digital zoom
- Video: Video Player/Recorder (MPEG4, H.263, H.264, WMV)
- Music: supports MP3, AAC, AAC+, WMA, ASF, WAV, WMV file types
- Images: supports JPEG, GIF, WBMP, BMP, MBM, PNG, D51TIFF/F, and animated GIF formats |
| Applications | - PIM: Contacts, calendar
- Office: Document Viewer (Word, Excel, PowerPoint, Adobe PDF, etc.) |



Sony Ericsson P910i**Category: High-end smart phones**

High-end smartphones are primarily PDA-type devices with phone capabilities. Examples include O2's XDA II, Palm's Treo 650, HP's iPaq series and Research in Motion's Blackberry 7100v.

- Display, data input**
 - Touch-screen, pen-based input (stylus, finger)
 - QWERTY keyboard on the inside of the flip
 - High-resolution LCD display
 - 208x320 pixel display with 262,144 colors
- Tri-band operation**
 - GSM 900/1800/1900; automatic switching between bands
- Data transfer**
 - GPRS
 - HSCSD (High-Speed Circuit-Switched Data)
 - TCP/IP
- Wireless connectivity**
 - Bluetooth wireless technology for data and audio connections
 - Over-the-air ringing tones, settings, and messages (for Web access point, Web bookmarks, and SyncML)
- Browsing**
 - Browser supporting WML (WAP 1.2.1)
 - CHTML, XHTML browser (WAP 2.0)
 - HTML and XHTML internet browser, supporting CSS and JavaScript
- Java**
 - Download applications and games
- Memory**
 - 64 MB RAM available for contacts, messages, ringing tones, images, video clips, PIM, and applications
 - Device supports up to 1 GB Memory Stick PRO Duo™
- Audio/visual media features**
 - Camera: VGA, 640x480 with programmable digital zoom
 - Video: playback and stream MPEG4, other codecs are also available
 - Music: supports MP3, AAC, RealAudio 7 and 8, WAV, MIDI, and AMR file types
 - Images: supports JPEG, GIF, WBMP, BMP, MBM, PNG, D51TIFF/F, and animated GIF formats
- Applications**
 - PIM: Contacts, calendar
 - Office: Quickword™ Quicksheet™ and Quickpoint™ viewers
 - Handwriting recognition
- Messaging**
 - Multimedia messaging: send and receive messages containing text, images, audio and video. Supports the 3GPP SMIL profile that allows you to send short presentations via MMS
 - Email: supports SMTP, POP3, and IMAP4 protocols
 - Text messaging: create, send, edit, and receive text messages (SMS)



Dell Axim X51v**Category: Palm-sized PDAs**

PDAs are handheld computers with voice capability and a range of sophisticated features that cannot be found on smartphones. The gap between smartphones and PDAs is blurring as shown by smartphones such as O2's XDA, MDA from T-Mobile, or Nokia Communicator 9300 and 9500 Series.

- | | |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Display | - 480 x 640 transmissive/reflective TFT color display supports more than 65,000 colors, landscape or portrait mode, active input area |
| | - 4 remappable application buttons, 5-way directional pad, touch screen |
| Data Transfer | - Built-in Bluetooth technology |
| | - IR port |
| Memory, processor | - 256 MB (160 MB internal flash drive, 55 MB program memory for applications and data) |
| | - 416MHz Intel® Xscale™ processor |
| Applications | - Palm OS® 5.4 |
| | - Expense, Note Pad, palmOne File Transfer, palmOne Quick Install, and palmOne Media3 (Windows only) |
| | - palmOne's enhanced PIM |
| | - Web browser and an email client (Blazer, VersaMail) |

**Samsung NEXiO XP40 Windows CE .NET PDA****Category: Handheld PCs**

Handheld PCs are portable computers that are small enough to be held in one's hand. They offer a wide range of applications but are not as convenient as notebooks because of their small keyboards and screens.

- | | |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Display | - 5 " transmissive TFT color LCD, 64,000 colors, resolution: 800 x 480, 190 dpi |
| | - full QWERTY keyboard |
| Data Transfer | - Integrated WiFi 802.11b |
| | - Bluetooth |
| | - Ethernet cards |
| | - Compact flash slot |
| Memory, processor | - Intel PXA270 520 MHz |
| | - 128 MB Flash ROM, 128 MB built-in RAM |
| Applications | - Windows CE .NET 5.0 operating system |
| | - Microsoft Office Viewers: Word, Excel, PowerPoint and Image Viewer. Pocket Office suite including Pocket Word, Microsoft PDF Viewer 2.1, Internet Explorer 5.5, and Pocket Outlook. |
| | - Terminal Services, MS Messenger, Windows Media Player, Voice Recorder as well as handwriting recognition. |
| | - 3rd party software: Flash Player 5, Jeode Java runtime, ActiveSync 3.6 |



Fujitsu LifeBook T4020

Category: Tablet PCs

Tablet PCs are notebook computers with LCD screens on which the user can write using a special-purpose pen, or stylus. Tablet PCs also typically have a keyboard and/or a mouse for input.



- | | |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Display | <ul style="list-style-type: none"> - 12.1-inch TFT XGA with 1024 x 768 resolution (up to 16.7M colors internal), 29.3 cm x 24.4 cm x 3.5 cm - Battery-free pressure pen with tether |
| Data Transfer | <ul style="list-style-type: none"> - Integrated 56K high speed modem and 10/100 NIC - IrDA 1.1 4Mbps, Bluetooth™ v1.2, 1000/100/10Mbps4 Gigabit Ethernet and Intel® PRO/Wireless 2915ABG network connection 802.11a/b/g with Dual Antenna |
| Memory, processor | <ul style="list-style-type: none"> - Standard 512MB DDR2 533MHz - Intel Pentium M 7501 (1.86GHz, 2MB L2 cache, 533MHz Front Side Bus) - 80-GB Hard Drive |
| Applications | <ul style="list-style-type: none"> - Microsoft® Windows® XP Tablet PC Edition - Adobe Acrobat® Reader v. 5.10, Microsoft Reader v. 2.5, Microsoft® Experience Pack for Tablet, Sun Java 1.4 |

Source: own research.

Appendix 2: ISO/IEC 9126 base measures

External Base Measures [ISO03]

Measure Name	Unit of Measurement
1. Number of functions	Function (number of)
2. Operation	Time (minutes)
3. Number of inaccurate computations encountered by users	Case (number of)
4. Total number of data formats	Format (number of)
5. Number of illegal operations	Operation (number of)
6. Number of items requiring compliance	Item (number of)
7. Number of interfaces requiring compliance	Interface (number of)
8. Number of faults	Fault (number of)
9. Number of failures	Failure (number of)
10. Product size	Byte
11. Number of test cases	Case (number of)
12. Number of breakdowns	Breakdown (number of)
13. Time to repair	Minute
14. Down time	Minute
15. Number of restarts	Restart (number of)
16. Number of restorations required	Restoration (number of)
17. Number of tutorials	Tutorial (number of)
18. Number of I/O data items	Item (number of)
19. Ease of function learning	Minute
20. Number of tasks	Task (number of)
21. Help frequency	Access (number of)
22. Error correction	Minute
23. Number of screens or forms	Screens (number of)
24. Number of user errors or changes	Error (number of)
25. Number of attempts to customize	Attempt (number of)
26. Total number of usability compliance items	Item (number of)
27. Response time	Second or millisecond
28. Number of evaluations	Evaluation (number of)
29. Turnaround time	Second or millisecond
30. Task time	Minute
31. Number of I/O-related errors	Error (number of)

Measure Name	Unit of Measurement
32. User waiting time of I/O device utilization	Second or Millisecond
33. Number of memory-related errors	Error (number of)
34. Number of transmission-related errors	Error (number of)
35. Transmission capacity	Byte
36. Number of revised versions	Version (number of)
37. Number of resolved failures	Failure (number of)
38. Porting user friendliness	Minute

Internal Base Measures [ISO03a]

Measure Name	Unit of Measurement
1. Number of functions	Function (number of)
2. Number of data items	Item (number of)
3. Number of data formats	Formats (number of)
4. Number of interface protocols	Protocol (number of)
5. Number of access types	Access type (number of)
6. Number of access controllability requirements	Requirement (number of)
7. Number of instances of data corruption	Instance (number of)
8. Number of compliance items	Item (number of)
9. Number of interfaces requiring compliance	Interface (number of)
10. Number of faults	Fault (number of)
11. Number of test cases	Test case (number of)
12. Number of restorations	Requirement (number of)
13. Number of input items which could check for valid data	Item (number of)
14. Number of operations	Operation (number of)
15. Number of messages implemented	Message (number of)
16. Number of interface elements	Element (number of)
17. Response time	Second or millisecond
18. Turnaround time	Second or millisecond
19. I/O utilization (number of buffers)	Buffer (number of)
20. Memory utilization	Byte
21. Number of lines of code directly related to system calls	Line (number of)
22. Number of I/O-related errors	Error (number of)
23. Number of memory-related errors	Error (number of)
24. Number of items required to be logged	Item (number of)
25. Number of modifications made	Modification (number of)
26. Number of variables	Variable (number of)
27. Number of diagnostic functions required	Function (number of)
28. Number of entities	Entity (number of)

Measure Name	Unit of Measurement
29. Number of built-in test function required	Function (number of)
30. Number of test dependencies on other system(s)	Dependency (number of)
31. Number of diagnostic checkpoints	Checkpoint (number of)
32. Number of data structures	Data structure (number of)
33. Total number of setup operations	Operation (number of)
34. Number of installation steps	Step (number of)

Quality in Use Base Measures [ISO04]

Measure Name	Unit of Measurement
1. Task effectiveness	(a given weight)
2. Total number of tasks	Task (number of)
3. Task time	Minute
4. Cost of the task	Dollar
5. Help time	Second
6. Error time	Second
7. Search time	Second
8. Number of users	User (number of)
9. Total number of people potentially affected by the system	Person (number of)
10. Total number of usage situations	Situation (number of)

Source: [ISO03; ISO03a; ISO04]

Appendix 3: CSS producing output similar to SSR technology

```
/* resizing the body to a smaller screen width */
body {
    max-width: 220px ! important;
    padding: 1px ! important;
    margin: auto ! important;
    border: groove black ! important;
}

/* setting font and line properties */
* {
    font-size: 11px ! important;
    padding: 1px ! important;
    text-align: left ! important;
    line-height: 1.05 em ! important;
}

/* changing document's settings */

/* canceling all size settings */
*:not(body):not(html){
    width: auto ! important;
    height: auto ! important;

/* width <= 220px */
    max-width: 220px ! important;

/* removing positioning */
    position: static ! important;

/* removing offsets */
    top: auto ! important;
    left: auto ! important;

/* removing floats */
    float: none ! important;

/* changing margins and paddings */
    padding: 0px ! important;
    margin: 0px ! important;

/* avoiding overflow on pre and table cells */
    white-space: normal ! important;
}

/* flattening all tables */
table, tbody, thead, tfoot, tr, td, th, col, colgroup {
    display: block ! important;
}

/* not displaying small images and adds */
```

```
img[width="1"], img[height="1"], img[width="468"], img[height="600"] {
    display: none ! important;
}

/* better placement of the bullet on a small screen */
li {
    list-style-position: inside ! important;
    display: inline !important;
}

/* getting rid of all iframes */
iframe {
    display : none ! important;
}

/* eliminating shockwave */
embed[type*="shockwave"] {
    display : none ! important;
}

div + a[href] {
    display: inline-block ! important;
    position: static ! important;
}

span + a[href] {
    display: inline-block ! important;
    position: static ! important;
}

/* changing anchors */
a[href] {
    text-decoration: underline ! important;
    color: blue ! important;
    display: inline-block ! important;
}

pre {
    white-space: pre-wrap ! important;
}
```

*Source: based on Daniel Glazman's proposal of a CSS for SSR,
http://daniel.glazman.free.fr/weblog/archived/2002_10_20_glazblogarc.html. Download: 2005-10-29.*

Appendix 4: UAProf for Nokia 7210

```

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschem-
20021212#" xmlns:mms="http://www.wapforum.org/profiles/MMS/ccppschem-20010111#">
<rdf:Description rdf:ID="Profile">
  <prf:component>
    <rdf:Description rdf:ID="HardwarePlatform">
      <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/-
        ccppschem20021212#HardwarePlatform" />
      <prf:BitsPerPixel>16</prf:BitsPerPixel>
      <prf:ColorCapable>Yes</prf:ColorCapable>
      <prf:CPU>ARM</prf:CPU>
      <prf:ImageCapable>Yes</prf:ImageCapable>
      <prf:InputCharSet>
        <rdf:Bag>
          <rdf:li>ISO-8859-1</rdf:li>
          <rdf:li>ISO-10646-UCS-2</rdf:li>
          <rdf:li>US-ASCII</rdf:li>
          <rdf:li>UTF-8</rdf:li>
        </rdf:Bag>
      </prf:InputCharSet>
      <prf:Keyboard>PhoneKeyPad</prf:Keyboard>
      <prf:Model>7610</prf:Model>
      <prf:NumberOfSoftKeys>2</prf:NumberOfSoftKeys>
      <prf:OutputCharSet>
        <rdf:Bag>
          <rdf:li>ISO-8859-1</rdf:li>
          <rdf:li>ISO-10646-UCS-2</rdf:li>
          <rdf:li>US-ASCII</rdf:li>
          <rdf:li>UTF-8</rdf:li>
        </rdf:Bag>
      </prf:OutputCharSet>
      <prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
      <prf:ScreenSize>176x208</prf:ScreenSize>
      <prf:ScreenSizeChar>15x6</prf:ScreenSizeChar>
      <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
      <prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
      <prf:TextInputCapable>Yes</prf:TextInputCapable>
      <prf:Vendor>Nokia</prf:Vendor>
      <prf:VoiceInputCapable>Yes</prf:VoiceInputCapable>
    </rdf:Description>
  </prf:component>
  <prf:component>
    <rdf:Description rdf:ID="SoftwarePlatform">
      <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
        ccppschem-20021212#SoftwarePlatform" />
      <prf:AcceptDownloadableSoftware>Yes</prf:AcceptDownloadableSoftware>
      <prf:CcppAccept>
        <rdf:Bag>
          <rdf:li>application/java-archive</rdf:li>
          <rdf:li>application/sdp</rdf:li>
          <rdf:li>application/vnd.nokia.ringing-tone</rdf:li>
          <rdf:li>application/vnd.oma.dd+xml</rdf:li>
          <rdf:li>application/vnd.oma.drm.message</rdf:li>
          <rdf:li>application/vnd.rn-realmedia</rdf:li>
          <rdf:li>application/vnd.symbian.install</rdf:li>
        </rdf:Bag>
      </prf:CcppAccept>
    </rdf:Description>
  </prf:component>
</rdf:Description>
</rdf:RDF>

```

```
<rdf:li>application/vnd.wap.wbxml</rdf:li>
<rdf:li>application/vnd.wap.wmlc</rdf:li>
<rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
<rdf:li>application/vnd.wap.xhtml+xml</rdf:li>
<rdf:li>application/x-x509-ca-cert</rdf:li>
<rdf:li>application/xhtml+xml</rdf:li>
<rdf:li>audio/3gpp</rdf:li>
<rdf:li>audio/aac</rdf:li>
<rdf:li>audio/amr</rdf:li>
<rdf:li>audio/amr-wb</rdf:li>
<rdf:li>audio/basic</rdf:li>
<rdf:li>audio/midi</rdf:li>
<rdf:li>audio/mp3</rdf:li>
<rdf:li>audio/mp4</rdf:li>
<rdf:li>audio/mpeg</rdf:li>
<rdf:li>audio/mpegurl</rdf:li>
<rdf:li>audio/rmf</rdf:li>
<rdf:li>audio/sp-midi</rdf:li>
<rdf:li>audio/wav</rdf:li>
<rdf:li>audio/x-au</rdf:li>
<rdf:li>audio/x-basic</rdf:li>
<rdf:li>audio/x-beatnik-rmf</rdf:li>
<rdf:li>audio/x-mpegurl</rdf:li>
<rdf:li>audio/x-pn-realaudio</rdf:li>
<rdf:li>audio/x-pn-realaudio-plugin</rdf:li>
<rdf:li>audio/x-rmf</rdf:li>
<rdf:li>audio/x-wav</rdf:li>
<rdf:li>image/gif</rdf:li>
<rdf:li>image/jpeg</rdf:li>
<rdf:li>image/jpg</rdf:li>
<rdf:li>image/png</rdf:li>
<rdf:li>image/tiff</rdf:li>
<rdf:li>image/vnd.nokia.ota-bitmap</rdf:li>
<rdf:li>image/vnd.wap.wbmp</rdf:li>
<rdf:li>image/x-bmp</rdf:li>
<rdf:li>image/x-epoc-mbm</rdf:li>
<rdf:li>image/x-wmf</rdf:li>
<rdf:li>multipart/mixed</rdf:li>
<rdf:li>text/calendar</rdf:li>
<rdf:li>text/css</rdf:li>
<rdf:li>text/html</rdf:li>
<rdf:li>text/plain</rdf:li>
<rdf:li>text/vnd.sun.j2me.app-descriptor</rdf:li>
<rdf:li>text/vnd.wap.wml</rdf:li>
<rdf:li>text/vnd.wap.wmlscript</rdf:li>
<rdf:li>text/x-co-desc</rdf:li>
<rdf:li>text/x-vCalendar</rdf:li>
<rdf:li>text/x-vCard</rdf:li>
<rdf:li>video/3gp</rdf:li>
<rdf:li>video/3gpp</rdf:li>
</rdf:Bag>
</prf:CcppAccept>
<prf:CcppAccept-Charset>
  <rdf:Bag>
    <rdf:li>ISO-8859-1</rdf:li>
    <rdf:li>ISO-10646-UCS-2</rdf:li>
    <rdf:li>US-ASCII</rdf:li>
```

```

        <rdf:li>UTF-8</rdf:li>
    </rdf:Bag>
</prf:CcppAccept-Charset>
<prf:CcppAccept-Encoding>
    <rdf:Bag>
        <rdf:li>base64</rdf:li>
        <rdf:li>quoted-printable</rdf:li>
    </rdf:Bag>
</prf:CcppAccept-Encoding>
<prf:CcppAccept-Language>
    <rdf:Seq>
        <rdf:li>en-US</rdf:li>
    </rdf:Seq>
</prf:CcppAccept-Language>
<prf:JavaEnabled>Yes</prf:JavaEnabled>
<prf:JavaPlatform>
    <rdf:Bag>
        <rdf:li>CLDC</rdf:li>
        <rdf:li>MIDP</rdf:li>
        <rdf:li>MIDP/1.0-compatible</rdf:li>
        <rdf:li>Profile/MIDP-2.0</rdf:li>
        <rdf:li>Configuration/CLDC-1.0</rdf:li>
    </rdf:Bag>
</prf:JavaPlatform>
<prf:JVMVersion>
    <rdf:Bag>
        <rdf:li>SunJ2ME1.0</rdf:li>
    </rdf:Bag>
</prf:JVMVersion>
<prf:Mexeclassmarks>
    <rdf:Bag>
        <rdf:li>1</rdf:li>
        <rdf:li>3</rdf:li>
    </rdf:Bag>
</prf:Mexeclassmarks>
<prf:MexecureDomains>No</prf:MexecureDomains>
<prf:OSName>Series60</prf:OSName>
<prf:OSVendor>Symbian LTD</prf:OSVendor>
<prf:OSVersion>2.1</prf:OSVersion>
<prf:RecipientAppAgent>BrowserMail</prf:RecipientAppAgent>
<prf:SoftwareNumber>7.0S</prf:SoftwareNumber>
<prf:VideoInputEncoder>
    <rdf:Bag>
        <rdf:li>H.261</rdf:li>
    </rdf:Bag>
</prf:VideoInputEncoder>
<prf:Email-URI-Schemes>
    <rdf:Bag>
        <rdf:li>pop</rdf:li>
        <rdf:li>imap</rdf:li>
        <rdf:li>http</rdf:li>
        <rdf:li>https</rdf:li>
    </rdf:Bag>
</prf:Email-URI-Schemes>
</rdf:Description>
</prf:component>
<prf:component>

```

```

<rdf:Description rdf:ID="NetworkCharacteristics">
<rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
          ccppschema-20021212#NetworkCharacteristics" />
  <prf:SupportedBluetoothVersion>1.0</prf:SupportedBluetoothVersion>
  <prf:SecuritySupport>
    <rdf:Bag>
      <rdf:li>SSL</rdf:li>
      <rdf:li>TLS</rdf:li>
    </rdf:Bag>
  </prf:SecuritySupport>
  <prf:SupportedBearers>
    <rdf:Bag>
      <rdf:li>GPRS</rdf:li>
      <rdf:li>CSD</rdf:li>
    </rdf:Bag>
  </prf:SupportedBearers>
</rdf:Description>
</prf:component>
<prf:component>
  <rdf:Description rdf:ID="BrowserUA">
  <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
          ccppschema-20021212#BrowserUA" />
    <prf:BrowserName>Nokia</prf:BrowserName>
    <prf:BrowserVersion>2.0</prf:BrowserVersion>
    <prf:FramesCapable>No</prf:FramesCapable>
    <prf:HtmlVersion>4.1</prf:HtmlVersion>
    <prf:JavaAppletEnabled>No</prf:JavaAppletEnabled>
    <prf:JavaScriptEnabled>No</prf:JavaScriptEnabled>
    <prf:PreferenceForFrames>No</prf:PreferenceForFrames>
    <prf:TablesCapable>Yes</prf:TablesCapable>
    <prf:XhtmlVersion>2.0</prf:XhtmlVersion>
    <prf:XhtmlModules>
      <rdf:Bag>
        <rdf:li>xhtml-basic10</rdf:li>
      </rdf:Bag>
    </prf:XhtmlModules>
  </rdf:Description>
</prf:component>
<prf:component>
  <rdf:Description rdf:ID="WapCharacteristics">
  <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
          ccppschema-20021212#WapCharacteristics" />
    <prf:SupportedPictogramSet>
      <rdf:Bag>
        <rdf:li>core</rdf:li>
      </rdf:Bag>
    </prf:SupportedPictogramSet>
    <prf:WapDeviceClass>C</prf:WapDeviceClass>
    <prf:WapVersion>2.0</prf:WapVersion>
    <prf:WmlDeckSize>357000</prf:WmlDeckSize>
    <prf:WmlScriptLibraries>
      <rdf:Bag>
        <rdf:li>Lang</rdf:li>
        <rdf:li>Float</rdf:li>
        <rdf:li>String</rdf:li>
        <rdf:li>URL</rdf:li>
        <rdf:li>WMLBrowser</rdf:li>
      </rdf:Bag>
    </prf:WmlScriptLibraries>
  </rdf:Description>
</prf:component>

```

```

        <rdf:li>Dialogs</rdf:li>
    </rdf:Bag>
</prf:WmlScriptLibraries>
<prf:WmlScriptVersion>
    <rdf:Bag>
        <rdf:li>1.2</rdf:li>
    </rdf:Bag>
</prf:WmlScriptVersion>
<prf:WmlVersion>
    <rdf:Bag>
        <rdf:li>1.3</rdf:li>
    </rdf:Bag>
</prf:WmlVersion>
<prf:WtaiLibraries>
    <rdf:Bag>
        <rdf:li>WTA.Public.makeCall</rdf:li>
        <rdf:li>WTA.Public.sendDTMF</rdf:li>
        <rdf:li>WTA.Public.addPBEntry</rdf:li>
    </rdf:Bag>
</prf:WtaiLibraries>
<prf:WtaVersion>1.1</prf:WtaVersion>
<prf:DrmClass>
    <rdf:Bag>
        <rdf:li>ForwardLock</rdf:li>
    </rdf:Bag>
</prf:DrmClass>
<prf:DrmConstraints>
    <rdf:Bag>
        <rdf:li>datetime</rdf:li>
    </rdf:Bag>
</prf:DrmConstraints>
<prf:OmaDownload>Yes</prf:OmaDownload>
</rdf:Description>
</prf:component>
<prf:component>
    <rdf:Description rdf:ID="PushCharacteristics">
        [...]
    </rdf:Description>
</prf:component>
<prf:component>
    <rdf:Description rdf:ID="MmsCharacteristics">
        [...]
    </rdf:Description>
</prf:component>
</rdf:Description>
</rdf:RDF>

```

Source: <http://nds.nokia.com/uaprof/N7610r100.xml>

Appendix 5: Questionnaire for the evaluation of MITL framework

PART A – General questions

A1)

What is your age and sex?

A2)

In which languages have you programmed before?

A3)

How would you grade your overall programming knowledge?

excellent (5)	very good (4)	good (3)	average (2)	weak (1)	non-existent (0)
<input type="checkbox"/>					

A4)

How many years of programming experience do you have?

< 1 year	1-2 years	2-3 years	3-5 years	5-8 years	> 8 years
<input type="checkbox"/>					

A5)

Are you familiar with:	Yes	No	Level*
1) HTML	<input type="checkbox"/>	<input type="checkbox"/>	_____
2) XHTML	<input type="checkbox"/>	<input type="checkbox"/>	_____
3) WML	<input type="checkbox"/>	<input type="checkbox"/>	_____
4) Java ME	<input type="checkbox"/>	<input type="checkbox"/>	_____
5) VoiceXML	<input type="checkbox"/>	<input type="checkbox"/>	_____
6) any other markup languages used for the development of mobile pages			

* For each language you know please enter your level of knowledge using the scale from the previous question (0 for non-existent, 5 for excellent knowledge).

PART B – General evaluation of MITL and Integrated Development Environment (IDE) for MITL

Please rate the MITL language and the Integrated Development Environment you used to complete the previous tasks. Please read each statement carefully. The evaluation scale is balanced, having an equal number of items for disagree and agree. The ends of the scale are typically extreme and therefore less frequently used. However, if you feel strongly about a question, please indicate so on the scale.

Scale: (1 for Strongly Disagree to 7 for Strongly Agree)

1. Strongly Disagree	2. Moderately Disagree	3. Mildly Disagree	4. Neither Disagree nor Agree	5. Mildly Agree	6. Moderately Agree	7. Strongly Agree
----------------------	------------------------	--------------------	-------------------------------	-----------------	---------------------	-------------------

Question	Answer
B1) MITL is powerful enough to handle most simple and some complex mobile user interfaces.	
B2) Given a choice between programming in a specific markup language (WML, XHTML, etc.) and MITL, I would choose MITL to build a simple GUI.	
B3) Given a choice between programming in a specific markup language (WML, XHTML, etc.) and MITL, I would choose MITL to build a complex GUI.	
B4) MITL is too complicated to use and I am discouraged from using it again.	
B5) Given what I know about Java and XML and my experience with MITL, I believe a declarative markup language can be as powerful as a traditional programming language when it comes to building user interfaces.	
B6) It was <i>easier</i> to build the GUI in MITL than it would have been if I had to build it in different markup languages.	
B7) It was <i>faster</i> to build the GUI in MITL than it would have been if I had to build it in different markup languages.	
B8) MITL can be used by non-professional programmers and occasional users.	
B9) MITL facilitates rapid prototyping of user interfaces.	
B10) MITL resembles other markup languages and I had no problems working with it.	
B11) The syntax of MITL is too limited to be used in real-world applications.	
B12) The development of applications based on MITL would be much more complicated without the Integrated Development Environment.	
B13) The IDE and its functionality are quite self-explanatory.	
B14) What is, in your opinion, the most important advantage of MITL? Answer:	
B15) What is the biggest drawback of this approach? Answer:	
B16) What are your suggestions for changes & improvements (MITL & IDE)? Answer:	
B17) What is your general impression? Answer:	

PART C - Usability and quality of MITL framework and its IDE

Please evaluate the MITL framework and the respective IDE with regard to the usability and quality of these products. Use the same scale as in part B of the evaluation.

Scale: (1 for *Strongly Disagree* to 7 for *Strongly Agree*)

1. Strongly Disagree 2. Moderately Disagree 3. Mildly Disagree 4. Neither Disagree nor Agree 5. Mildly Agree 6. Moderately Agree 7. Strongly Agree

Question	Answer
C1) Learnability, understandability: I was able to understand the concept behind JSP Tag Libraries.	
C2) Learnability, understandability (MITL): MITL is easy to learn and apply.	
C3) Learnability, understandability (JSTL): JSTL is easy to learn and apply.	
C4) Learnability, understandability (Web Language): Web Language is easy to learn and apply.	
C5) Learnability, understandability (RSS libraries): RSS libraries are easy to learn and apply.	
C6) Likeability: I like the concept of MITL.	
C7) Accuracy: The results produced with the help of the MITL library were correct and did not differ from my expectations.	
C8) Resource utilization: The MITL IDE is resource-intensive.	
C9) Attractiveness: The GUI of the MITL IDE looks attractive to me.	
C10) Likeability: I like the product (the MITL library and its IDE).	
C11) Flexibility: The IDE can be tailored to suit my personal preferences (look and feel of the application, choice of additional emulators, etc.)	
C12) Minimal Action: With MITL and its IDE, the development of device-independent applications can be achieved in a minimal number of steps.	
C13) Minimal Memory Load: With MITL and its IDE, the development of device-independent applications can be achieved while keeping a minimal amount of information in mind.	
C14) User Guidance: The IDE provides context-sensitive help and meaningful feedback when errors occur.	
C15) Self-Descriptiveness: The IDE gives clear assistance and supports user's operations.	
C16) Completeness: I was able to program a device-independent application in MITL and complete my task with the help of the IDE.	

Question	Answer
C17) Simplicity, Familiarity, Readability: It was easy to understand the content of text dialogs and visual elements in MITL IDE.	
C18) Controllability: I had the feeling that I am in control of the IDE while programming with its help.	
C19) Navigability: It was easy to navigate in the IDE in an efficient way.	

PART D – MITL-based approach and Device Independence Principles

Evaluate the MITL-based approach considering the *Device Independence Principles (DIPs)*. For each principle write whether MITL fulfills it (strongly agree) or totally ignores it (strongly disagree).

Scale: (1 for *Strongly Disagree* to 7 for *Strongly Agree*)

1. Strongly Disagree 2. Moderately Disagree 3. Mildly Disagree 4. Neither Disagree nor Agree 5. Mildly Agree 6. Moderately Agree 7. Strongly Agree

Principle	Evaluation
D1) DIP-1: Device-independent access	
<input type="checkbox"/> For some Web content or application to be device independent, it should be possible for a user to obtain a functional user experience associated with its Web page identifier via any access mechanism .	
D2) DIP-2: Device-independent Web page identifiers	
<input type="checkbox"/> A Web page identifier that provides a functional user experience via one access mechanism should also provide a user experience of equivalent functionality via any other access mechanism.	
D3) DIP-3: Functionality	
<input type="checkbox"/> It should be possible to provide a functional user experience , in response to a request for a Web page, in any given delivery context that has an adequate access mechanism.	
D4) DIP-4: Incompatible access mechanism	
<input type="checkbox"/> If a functional user experience of an application cannot be provided due to inherent limitations in the access mechanism, an explanatory message should be provided to the user.	
D5) DIP-5: Harmonization	
<input type="checkbox"/> If the author wishes, it should be possible to provide a harmonized (well-designed) user experience, in response to a request for a Web page, in any given delivery context that has an adequate access mechanism.	

Part E – MITL framework and Device Independence Authoring Challenges

Please read a short summary of Device Independence Authoring Challenges and evaluate the MITL-based approach with regard to them. Use the following scale:

Scale: 1: non existing; 2: low; 3: middle; 4: high; 0: does not apply to this method

Approach Challenge/ Evaluation criteria	Evaluation of MITL
Provide comprehensive scope	
<p>DIAC-3.1: Application scope The approach is able to support all types of Web applications for all devices.</p>	
<p>DIAC-3.19;3.20: Integration of device-dependent and independent content With MITL it is possible to integrate external, device-dependent and device-independent content into existing content.</p>	
<p>DIAC-3.28: Range of complexity MITL supports simple and complex applications.</p>	
Support smooth extensibility	
<p>DIAC-3.2: Extensible capabilities MITL is extensible and allows extending this technique to new technologies.</p>	
<p>DIAC-3.29: Scalability of complexity Increasing application complexity implies rising complexity of a JSP page with MITL elements.</p>	
Support simplicity	
<p>DIAC-3.4: Simplicity Simple applications can be developed without great effort; the complexity of development scales smoothly with the complexity of applications</p>	
<p>DIAC-3.11: Simple content MITL enables the use of the simplest forms of content</p>	
Support delivery context variability	
<p>DIAC-3.5: Navigation variability Various navigation methods are supported by MITL and they can be implemented with minimal effort.</p>	
<p>DIAC-3.6: Organization variability The amount of content displayed on various devices with different presentation techniques can vary in MITL.</p>	
<p>DIAC-3.7: Media variability On different devices different media types are supported if necessary (e.g. audio, video, etc.).</p>	

Approach Challenge/ Evaluation criteria	Evaluation of MITL
Support author specified variability	
DIAC-3.12: Text content variety Different versions of text are supported and displayed depending on the device.	
DIAC-3.13: Media resource variety Various media formats are supported according to the capabilities of devices.	
DIAC-3.14;3.15: Media resource specification & selection Authors are able to define lists of media types for different delivery contexts. MITL enables the selection of a resource from a list of possibilities.	
DIAC-3.30;3.31: Aggregation & decomposition Aggregation of presentation units is supported. Decomposition of resources is possible.	
DIAC-4.3: Layout variety Different spatial and temporal layouts are supported for different delivery contexts.	
Affordability	
DIAC-3.3: Affordability (cost) The development of device-independent applications does not require excessive effort or high investments	
DIAC-3.33: Reusing existing applications MITL provides support for including parts of existing applications during the design or at runtime.	
DIAC-6.5: Minimization of effort Support for a variety of devices is possible without excessive effort.	
DIAC-6.13: Separation of device-dependent and device-independent material Device-independent material can be separated from device dependent material.	
DIAC-6.6: Abstraction of device knowledge Detailed knowledge about the target devices is not required.	
DIAC-6.10: Scalability of effort/quality The author is able to decide how much effort he/she would like to invest in the customization process.	

Appendix 6: Questionnaire for the evaluation of MoWeSA framework

PART A – Web Services Tag Library

Please rate the WSTL you used to complete the previous task. Please read each statement carefully. The evaluation scale is balanced, having an equal number of items for disagree and agree. The ends of the scale are typically extreme and therefore less frequently used. However, if you feel strongly about a question, please indicate so on the scale.

Scale: (1 for *Strongly Disagree* to 7 for *Strongly Agree*)

1. Strongly Disagree 2. Moderately Disagree 3. Mildly Disagree 4. Neither Disagree nor Agree 5. Mildly Agree 6. Moderately Agree 7. Strongly Agree

Question	Answer
A1) WSTL is powerful enough to support all types of Web Services.	
A2) Given a choice between invoking Web Services calls in a programming language (Java, VB.Net , etc.) and in WSTL, I would choose WSTL to build a simple GUI based on Web Services.	
A3) Given a choice between invoking Web Services calls in a programming language (Java, VB.Net , etc.) and in WSTL, I would choose WSTL to build a complex GUI based on Web Services.	
A4) WSTL is too complicated to use and I am discouraged from using it again.	
A5) It was easier to communicate with Web Services using WSTL than it would have been if I had to apply a programming language.	
A6) It was faster to communicate with Web Services using WSTL than it would have been if I had to apply a programming language.	
A7) WSTL can be used by non-professional programmers and occasional users.	
A8) WSTL facilitates rapid prototyping of mobile Web Services.	
A9) WSTL resembles markup languages and I had no problems working with it.	
A10) The syntax of WSTL is too limited to be used in real-world applications.	

PART B - Usability and quality of MoWeSA framework and its IDE

Please evaluate the MoWeSA frameworks, its elements and IDE with regards to the usability and quality of these products. Use the same scale as in part A of the evaluation.

Scale: (1 for *Strongly Disagree* to 7 for *Strongly Agree*)

1. Strongly Disagree 2. Moderately Disagree 3. Mildly Disagree 4. Neither Disagree nor Agree 5. Mildly Agree 6. Moderately Agree 7. Strongly Agree

Question	Answer
B1) Learnability, understandability (WSTL): WSTL is easy to learn and apply.	
B2) Likeability: I like the concept of WSTL and the MoWeSA framework.	
B3) Accuracy: The results produced with the help of the WSTL library were correct and did not differ from my expectations.	
B4) Resource utilization: The MoWeSA IDE is resource-intensive.	
B5) Likeability: I like the product (the MoWeSA framework and its IDE).	
B6) Minimal Action: With the MoWeSA framework and its IDE, the development of device-independent applications based on Web Services can be achieved in a minimal number of steps.	
B7) Minimal Memory Load: With the MoWeSA framework and its IDE, the development of device-independent applications based on Web Services can be achieved while keeping a minimal amount of information in mind.	
B8) Completeness: I was able to program a device-independent application in MoWeSA and complete my task with the help of the IDE.	
B9) Simplicity, Familiarity, Readability: It was easy to understand the content of text dialogs and visual elements in the IDE supporting MoWeSA .	
B10) What is, in your opinion, the most important advantage of the MoWeSA framework?	Answer:
B11) What is the biggest drawback of this approach?	Answer:
B12) What are your suggestions for changes & improvements (WSTL & IDE)?	Answer:
B13) What is your general impression?	Answer:

Ehrenwörtliche Erklärung

Hiermit erkläre ich ehrenwörtlich, dass ich bisher an keiner Doktorprüfung teilgenommen habe. Ferner versichere ich ehrenwörtlich, dass ich die vorliegende Abhandlung selbst verfasst, mich keiner fremden Hilfe bedient und keine anderen als die im Schriftenverzeichnis der Abhandlung angeführten Schriften benutzt habe. Die Abhandlung war und ist nicht Gegenstand einer Doktorprüfung an einer anderen Universität, Hochschule oder Fakultät.

Bożena Jankowska

London, den 03.11.2006

CURRICULUM VITAE

BOŻENA JANKOWSKA

Born: July 04 1978, Poznań, Poland

Citizenship: Polish

EDUCATION

11/2002 – 11/2006

European University Viadrina (Frankfurt/Oder, Germany)

Ph.D. student , supervised by Prof. Dr. E. Stickel

Dissertation: Architectural frameworks for automated content adaptation to mobile devices based on open source technologies

Awarded scholarship from the German government

1997 – 2002

European University Viadrina (Frankfurt/Oder, Germany)

M.A., Business Administration

Majors: Banking and Finance, Business Informatics

Final grade: 1.6

Diploma thesis: Development of a Demo-Application for Measuring Efficiency in the German Banking Sector Using Data Envelopment Analysis.

1995 – 1996

Midview High School (Grafton, Ohio)

1993 – 1997

Maria Magdalena High School (Poznań, Poland)

Baccalaureate with distinction

WORK EXPERIENCE

04/2006 – present

Business Objects (London, United Kingdom)

Senior Consultant, Business Intelligence

02/2005 – 03/2006

Hewlett Packard, Global Technology Solutions Group (Warsaw, Poland)

Business Intelligence specialist

04/2002 – 11/2002

Chair of Business Informatics, Prof. Dr. Karl Kurbel European University Viadrina (Frankfurt/Oder, Germany)

Student assistant

03/2001 – 06/2001

arsmovendi.com AG (München, Germany)

Trainee in the IT department

08/2000 – 09/2000

NetFederation Interactive Media GmbH (Köln, Germany)

Trainee in the IT department

04/1999 – 07/1999

Chair of Business Informatics, Prof. Dr. Karl Kurbel European University Viadrina (Frankfurt/Oder, Germany)

Student assistant