CONTENT ADAPTATION FOR HETEROGENEOUS MOBILE DEVICES

Robert Schmohl
Institut für Informatik
Technische Universität München
Boltzmannstr. 3
85748 Garching bei München
Germany
Email: schmohl@in.tum.de
Phone: +49-89-289-18566
Fax: +49-89-289-18557

Uwe Baumgarten
Institut für Informatik
Technische Universität München
Boltzmannstr. 3
85748 Garching bei München
Germany
Email: baumgaru@in.tum.de
Phone: +49-89-289-18564
Fax: +49-89-289-18557

Lars Köthner
Comnos GmbH
Nymphenburger Str. 20a
80335 München
Germany
Email: lars.koethner@comnos.de
Phone: +49-89-2000329-0
Fax: +49-89-2000329-99

ABSTRACT

Recent advances in mobile computing have spawned a very heterogeneous environment of mobile devices, which is reflected by the presence of the devices' different capabilities. This chapter focuses on handling this device heterogeneity in the context of content adaptation of mobile services, so that generic content can be provided to any device in the heterogeneity spectrum. We present an approach, which enables mobile services to adapt its content provision to a mobile device by considering the device's content provision capabilities. Those capabilities encompass both the communication channels for content delivery, and the capabilities to present content to the user. Our approach is designed as a service platform, which implements a content adaptation procedure for web-based mobile services by utilizing device capability databases and generic page transformation. This approach enables mobile devices to visualize any generic content device-specifically on their integrated browsers.

INDEX WORDS

- Mobile computing
- Mobile services
- Mobile device heterogeneity
- Content adaptation
- Multi-channel service provision
- Web-based platform
- XML transformation

CONTENT ADAPTATION FOR HETEROGENEOUS MOBILE DEVICES

INTRODUCTION

Since mobile computing is getting increasingly popular, the development of mobile services is getting increasingly complex implying new challenges to be handled (Schmohl, 2007; Want, 2005). One of those challenges is the highly heterogeneous environment of mobile devices, which has emerged as a consequence of the rapid mobile computing evolution of the past years. Most companies handle the heterogeneity of mobile devices by employing the Pareto principle, also known as the 80-20-rule. In this context, this approach proposes to make a mobile service available for 80% of the users, which employ 20% of devices available on the specific target market. However, although this approach may work in practice, it lacks scalability and it requires a high level of maintenance implied by the constantly ongoing evolution of technologies. Hence, this solution of the problem is temporary at best.

Mobile devices' heterogeneity can be divided into hardware and software heterogeneity (da Rocha, 2005). Hardware heterogeneity reflects the presence of devices with different capabilities. Software heterogeneity describes the presence of different operating systems and applications running on mobile devices. Speaking of the provision of web-based mobile services, we face both hardware and software heterogeneity, which is influenced by the following aspects:

- *Markup languages:* Mobile devices support several different markup languages to display output. However, most of them support only a few, so delivery of output is highly dependent on the target device's supported markup language.
- *Device output capabilities:* Devices have very different capabilities in processing output visually and acoustically (see the subsequent section about content adaptation).
- *Logical communication channels:* Mobile devices' communication is mapped on physical and logical channels. While a device's physical communication channels are intended to stay transparent to both the user and the mobile service, the awareness of logical channels does matter. From this perspective, service requests and provision occur on logical channels (such as SMS, MMS, E-mail, voice, etc.), whose availability is completely device-dependent.

To tackle those heterogeneity issues we have developed a concept of a web-based platform for mobile services, which handles both content adaptation and multi-channel service provision. The basic idea behind this concept is based on a single request-response dialog between the user of the mobile service and the platform providing it. The outline of this concept encompasses the creation of a generic and device-independent content-page, which is adapted to the requesting device using a device capability database and XML transformation techniques. The conceptual design of the web-based platform introduced here includes multi-channel communication and modular configuration (service creation, discussed further on in this chapter).

While the transformation of generic content covers the heterogeneity issues concerning devices' markup languages and output capabilities, the multi-channel communication aspect enables the provision of services on different logical channels, thus handling the correspondent remaining heterogeneity issue listed above. The aspect of configuring the platform modularly aims at omitting the need of creating services by means of programming. We propose to assemble mobile services out custom building blocks, which are individually *modeled* instead of programmed. This aspect results in significant simplification of service creation, since the complexities of both the service creation and the underlying web framework are reduced.

The conceptual design introduced here has been developed in a joint effort by the Technische Universität München and the Comnos GmbH to complement an existent mobile service platform, the *Open Dialog Platform (ODP)* (Comnos GmbH, 2007). The ODP is capable of providing services by SMS, MMS and email, so that the work presented in this chapter aims at complementing the ODP by enabling service provision on the web channel. We have validated this conceptualization by implementing the web-channel-platform accordingly. We are going to discuss the implementation after the conceptual discussion in the subsequent sections.

The rest of this chapter is structured as follows: In the next section we discuss the aspects relevant for a web-platform, which aims at providing mobile services. Those aspects reflect an as-is analysis of current technologies. Afterwards, we present an overview about the utilized concept and the corresponding platform design. We carry on with the discussion of our approach to solve the heterogeneity problem. With the concept introduced, we briefly present our implementation of the platform before we subsequently conclude this chapter by summarizing our work and discussing the outlook on our future work.

BACKGROUND

To determine the relevant aspects for our approach, we have evaluated a set of Java-based web frameworks. The reason for putting our focus on this type of frameworks is the setting of our current system (Comnos GmbH, 2007), which runs in a J2EE environment and which is intended to host our mobile service platform implementation. We are going to discuss the implementation later on. At this moment, we put our focus on the web framework evaluation, which we have conducted to identify the current state-of-the-art techniques in web framework development.

The evaluation has encompassed frameworks developed by the Apache Foundation (Apache Software Foundation, 2007), such as Struts, Turbine, Tapestry, Velocity and Cocoon, as well as Spring (Spring, 2007), WebWork (Open Symphony, 2007) and Java Server Faces (Sun Microsystems Inc., 2007). The following enumeration shows the aspects, that we have identified during the evaluation and which we consider as most relevant for a web framework employed for our approach:

- *Request mapping:* Those aspects describe how incoming client requests are handled. This primarily denotes which URLs are mapped to which application entry points handled by the web framework. It also includes the issue of assigning existent sessions to its users. The evaluation has shown that XML is a widely accepted method in configuring request mappings.
- *Page flow control:* This aspect consists of the sequence of pages, which a user can request in a web application. A logical abstraction of this aspect is to describe all possible sequences of pages by a directed graph, denoting the

page flow. For this purpose, XML is a reasonable format to represent such a data structure in this case, too.

- *Business logic execution:* The business logic denotes all logic necessary to calculate results, triggered by requests and influencing the content returned to clients. The web framework controls its execution, which occurs in the period between receiving a client request and issuing the appropriate response. Business logic is usually encapsulated in Java Beans or POJOs ("Plain Old Java Objects"), loaded by the web framework and executed as *actions*.
- *Output rendering:* The actual construction of the output data, which is subsequently sent back to the client, is the main part of this aspect. Since we are dealing with a highly heterogeneous set of possible clients, the device-specific adaptation of output has to be handled adequately. A common approach is to use generic markup to describe the output's static content and to create the final output with regard to dynamic content and device data at runtime. Since we focus on mobile devices, the output needs to be delivered to fit on adequate display sizes.
- *Configurability:* This aspect encompasses the configuration of all dynamic functionalities of a framework. This may include output page content, page flow definitions, etc. Configurations are to be modular and adequately represented in order to be easily deployable.
- *Definition of services*: As a consequence of our above definition of configurations we can state, that a user-accessed web-based service is represented by the set of configurations interpreted by the framework. Since our focus is set on *mobile* services, we especially emphasize the issues of creating proper output pages tailored for mobile devices. This encompasses the minimization of content and the creation of device-specific layout.
- *Stability and performance*: Web frameworks are required to handle large amounts of simultaneous requests. For this reason, non-functional aspects, such as stability and performance, are equally important.

After having evaluated the web frameworks, we have come to the conclusion, that none of them is suitable to be employed as a base for our concept. The reason is that every framework has a different strong emphasis on each of the various listed aspects. JSF and Spring, for example, handle the aspect of defining page flows very neatly. Cocoon, on the other hand, is doubtlessly the most powerful framework when it comes to adapting content to a large amount of different formats. However, Cocoon lacks the strengths of other frameworks and integrates poorly into modular systems. It also performs just moderately. For those reasons, we decided not to employ one of the evaluated frameworks neither as the controlling component nor as subordinate modules. Our consideration of employing Cocoon was put aside due to the integration issues.

For the just discussed reasons, our course of action has been to regard the aspects listed above as a frame for the conceptualization of our own web platform, which we present in this chapter. Hence, these aspects must be refined so that they can be interpreted as basic requirements, which need to be met in the platform design. It is discussed in the next section.

PLATFORM DESIGN

With the frame of requirements identified, we now introduce the design of the platform implementing our approach. We start exploring the platform's configuration modules, which depict an important cornerstone of our concept. Afterwards, we describe the workflow illustrating the basic working principle of providing content. We close this section by providing a brief architectural draft, which reflects both the workflow and the configuration modules.

Configuration

Our platform proposal utilizes modular configuration with the individual configuration modules used as building blocks for the complete definition of a mobile service. There are 4 types of configuration modules:

- *Page flow definition:* defines the page flow of the service. It represents a graph with its edges denoting output pages and business logic actions, and its vertices denoting the transitions to interconnect them according to the service's specification.
- *Business logic definition:* defines the business logic executed during page flow traversing. This particularly includes the transition of communication channels within a mobile service execution. The single steps to be executed are summarized in a *modeled* definition, which is interpreted and executed by the platform. In addition to modeling business logic, regularly *programmed* libraries can be loaded and executed by the platform. However, a detailed discussion of this topic is out of the scope of this chapter.
- *Page definition:* defines the structure and content of an output page. We have identified several basic page types, which page definitions can be exclusively assigned. Those page types enable a structured subset of all reasonable pages modelable with markup languages. They cover most use cases, such as representing content, input forms or result lists, for example. The constraint of page definitions being assigned a single page type out of a limited amount of page types greatly reduces the complexity, which is normally given when using markup languages. Hence, the complexity of defining page definitions is reduced without significantly limiting their expressional power.
- *Style definition:* defines the stylistic properties of output pages. This may include layout, colors, font sizes, etc.

All of those configuration types are instantiated as XML-defined configurations, so that they can be read and processed by both men and machines. The coherence of those configurations with our platform proposal is discussed later on in this chapter.

Figure 1 visualizes in which context those configurations are settled. The main purpose for the modular conception of those configurations is to omit the need to define them by means of programming (as stated in the introductory section). Defined sets of those configurations represent the execution manual for the service platform to provide a specified service, so that those sets can basically be interpreted as the service definitions themselves. This approach reduces the complexity of the service definition, the platform design and the handling of heterogeneity aspects for the following reasons:

- *Complexity reduction of service definition:* A mobile service is *modeled*, instead of being programmed. The designer of a mobile service has a structured set of design items (business logic, page types, style types) available for modeling, which cover most use cases.
- *Complexity reduction of platform design:* The platform simply loads and utilizes the configuration modules. This makes the platform scalable and easy to maintain.
- *Complexity reduction of device heterogeneity:* Device heterogeneity does not need to be handled in configurations. Instead, the platform's dedicated components accomplish this task.

A notable consequence of complexity reduction and the XML-based modeling approach is the minimization of effort to create services.

Workflow

The workflow discussed here describes a typical dialog between the mobile device (the client) and our service-provisioning platform (the server). Such a dialog is also generally referred to as a *request-response-cycle*.

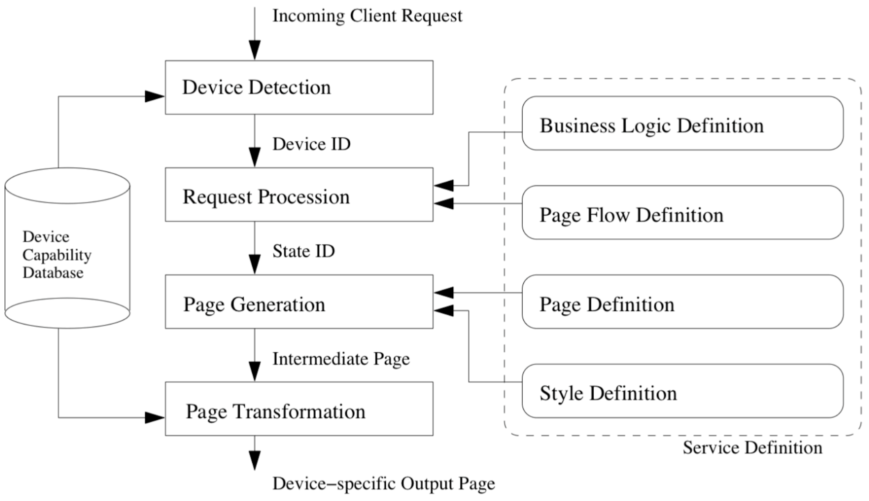

Figure 1 - Workflow

Device detection An incoming request from a mobile device is received by the platform via the HTTP protocol (HTTP, 2007). The *user-agent* header from this request is extracted and looked up in a device database to identify the requesting device. With the device identified, all device capabilities are retrieved from the database as well. This *device capability database* (DCDB) contains all relevant and device-specific information to accurately render output pages for display at the respective client device.

Request procession The base of processing client request is the concept of *page flows*, which denotes a sequence of states traversed by the client. Those flows are mapped on the page flow definition, which defines all possible flows. Graphically represented, a single page flow is a directed path in the page flow definition graph. The page flow definition of a web-based service includes dedicated states for output pages and business logic (vertices in the graph). Those states are connected by directed transitions (edges in the graph). Put in sequence, those states create a flow, which basically denotes the workflow of a service. A user of a service only recognizes the transition between an output-state to another, since the output pages assigned to the correspondent output-state are shown to the user. The transition between two output-states triggers all business logic defined in the business-logic-states on the path between those two output-states. This approach allows the construction of services, which symbiotically combine both output and business logic. Figure 2 shows an exemplary page flow definition. A flow in the scenario described in this figure is to be interpreted as the login process conducted by a user.

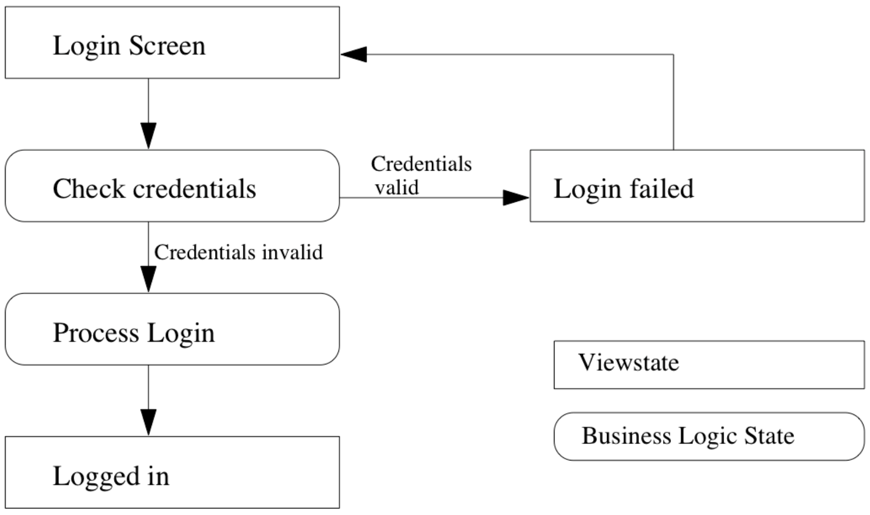

Figure 2 - Example of a page flow definition

Intermediary page generation Before being returned to the client the output information is first generated as a device-independent meta-page. It serves as an *intermediary* representation of all the content, which is supposed to be on the final device-specific output page. The intermediary page is generated from a static page definition and a style definition, which both have been identified after successfully conducting the step to the output-state in the page flow. The generation process consists of 2 independent tasks:

- *Fusion of content and style:* The page- and style definition are concatenated, so that the intermediary page includes information about both style and content.
- *Dynamic reference dissolving:* In addition to static content, both page- and style definition may contain references to dynamic data, which is available at runtime only. This includes data defined by the execution of business logic, data passed by the user in the request, etc. During the generation of the intermediary page, those references are dissolved and the current data is fetched from the mobile service's current context.

The resultant intermediary page includes all of the content information to be sent to the client device. As the style- and page definitions, it is a valid XML document. It is to be emphasized, that it is still lacking device-specifity and therefore needs to be adapted. Figure 3 visualizes the intermediary page generation.

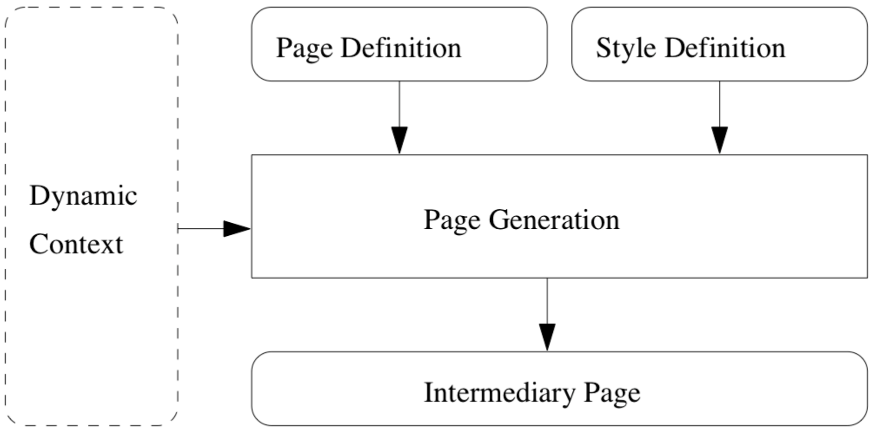

Figure 3 - Page generation process

Page transformation and content adaptation In order to be displayed properly by the client device the intermediary page is transformed according to the client's device-specifications. Since the intermediary page and the final output page are valid XML documents (the markup language of the output page is an XML-derivate) this is done using parameterized XSL-T (XSL-T, 2007), a common XML transformation technique. The adaptation process includes the conformance to the device's support of both markup language and media. To do so, the process furthermore includes the utilization of the device data extracted from the DCDB in the beginning of the request-response-cycle. Hence, the adaptation process consists of 2 concurrently handled aspects:

- The transformation of the intermediary page into a page complying with the target device's markup language support
- The adaptation of media to be displayed properly on the target device's screen

The resultant output page is subsequently returned to the client, completely tailored to its device capabilities.

At this point, the reader is advised to re-review Figure 1, which visualizes the workflow discussed in this section.

Architectural Draft

The architectural draft presented here strongly reflects the workflow described in the previous section. The platform is composed of 3 basic components:

- *Device detector:* This component handles the device detection and the connection to the DCDB. It is responsible for loading device data into the platform.
- *Request processor:* This building block handles web-specific procession of incoming client requests, such as request mapping and page flow procession.
- *Output generator:* The generation of the intermediary page and its adaptation to the client device is realized by this component.

Those components are structured in 2 layers: the presentation layer handling device detection and content adaptation, and the core framework handling the request procession. Figure 4 visualizes this proposed architecture.

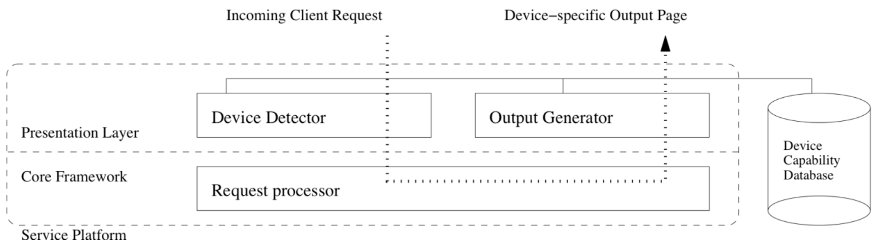

Figure 4 - Architectural draft

CONTENT ADAPTATION CONCEPT

This section discusses the adaptation of the output data to meet the client device's specifications. First, we decompose the mobile devices' heterogeneity by structuring the device-specific information. Then, we have a look on the working principle of the content adaptation process, which is composed of two subordinate tasks: the transformation of the intermediary page and the adaptation of all containing media content in the transformed page.

Handling Device Heterogeneity

The heterogeneous capabilities of mobile devices regarding the visualization of content received while accessing a web-based service have been abstracted as follows:

- *Supported markup language:* The mobile device's integrated browser supports a specific set of markup languages. However, even though markup languages are standardized, some manufacturers extend their devices' markup support by adding their own specifications. All markup support is based on the browser employed by a mobile device, which can be a device's manufacturer's design (Nokia, for example, employs its own browsers in its devices) or an external product, such as the externally licensed OpenWave Browser (Openwave Systems Inc., 2007). Hence, markup support is not dependent on devices, but on the mobile devices' browsers.
- *Media output capabilities:* Mobile devices have different capabilities to output both visual and acoustical media. Those capabilities may include physical display properties, supported media file formats, etc. For example, a newer mobile device may support colored Jpeg images whereas an older device may only be able to display WBMP (WBMP, 2007) images on a monochrome display. Depending on such capabilities, media content has to be delivered in the particular format meeting the target device's capabilities.
- *Supported logical channels:* As stated before, mobile services can be provided on numerous channels, such as SMS, MMS, voice and/or web. Since the work described in this paper focuses on the web channel, we state this aspect on account of completeness and we briefly discuss it later on.

Based on this abstraction of the heterogeneity faced while considering our content adaptation approach the adaptation process can be decomposed into the following two steps, which are discussed in the subsequent sections:

1. The transformation of the generic intermediary page into the output page composed in the target device's supported markup language.

2. The adaptation of all media content meeting the target device's specification concerning the visual and acoustical output.

XSL-T Transformation and Parameterization

As outlined during the introduction of the workflow, the content adaptation process starts when the intermediate page is constructed. Since both the intermediate page and the adapted output page are valid XML documents, the intermediate page is transformed into the device-specific page using XSL-T. To incorporate the device specifics into the content adaptation procedure the device data, which has been determined during the device detection phase, is used to parameterize the transformation process. That procedure can be decomposed into the following steps as visualized in Figure 5:

1. *Stylesheet selection:* XSL-T transformations are defined by stylesheets, which include all transforming instructions. In our approach we propose a stylesheet for each markup language. Since the device is identified at this point of the workflow, the device's browser - and therefore its supported markup language - is known and hence specifies the stylesheet for the transformation. If the browser supports several markup languages a prioritized selection is made.
2. *Inclusion of device parameters:* Since the interpretation of markup is not language-specific, but browser-specific, certain device characteristics may be of importance to display the output page properly at the client device. That's why all relevant device parameters are passed as XSL-T parameters to the transformation engine.
3. *Transformation:* With the stylesheet of the target markup language selected and all device parameters known, the transformation process is conducted outputting the device-specific page.

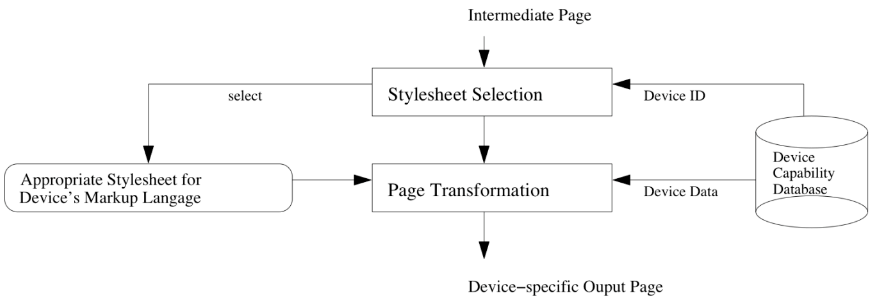
Intermediate Page
select
Stylesheet Selection
Device ID
Device Capability Database
Appropriate Stylesheet for Device's Markup Langage
Page Transformation
Device Data
Device−specific Ouput Page

Figure 5 - Transformation process

At this point, we exploit the reduction of complexity of markup languages, which we argued earlier while introducing our modular configuration concept. The page- and style definitions are minimized to a few types, thus being manageable and far less complex than original markup equivalents (i.e. HTML). Hence, the intermediary page inherits that advantage and we exploit the fact that a manageable intermediary XML-page is easier to transform, than a document in a standardized markup language (i.e. HTML, again).

Adaptation of Media Content

While having abstracted the device heterogeneity aspects we have stated that the output of media relies heavily on device capabilities. For this reason, we need to know what the requesting client device is actually capable of. This information has already been made available by the device detector at the beginning of the request-response-cycle (see previous section about the workflow) and it is passed in the form of parameters to the transformation engine prior to the transformation.

The intermediate page includes references to original media items, which are all device-independent. Although, the intermediary page generation includes the dissolving of all dynamic references, the dissolving of media references has been explicitly postponed, since the intermediary page is supposed to be device-independent. Instead, the page transformation substitutes all references to original media, so that the output page only references device-specific media, which can be outputted properly at the client device.

Those references to device-specific media imply an existent repository with all imaginable device-specific derivates of all available (original) media items. This repository needs to be structured by the heterogeneity aspects defining the output capabilities of mobile devices. In particular, we propose the following criteria for such a structuring:

- *Type of media:* This may include the basic type of a media element, such as image or sound.
- *Type of usage:* Another structural criterion may be the information on how the media element is to be used. If we consider an image, the type of usage may be “thumbnail”, “preview”, etc.
- *Media Format:* Electronic media is available in an abundance of formats. Regarding the example with images again, there are quite a few broadly supported formats, such as Jpeg, GIF, PNG, etc. Hence, structuring of media formats seems reasonable.
- *Media properties:* This structuring criterion denotes properties of format-independent media elements, such as minimum size when speaking of the image example again.

The media repository may be structured following the criteria proposed above. More precisely, this means, that each media element corresponds to a set of all possible derivates fitting the aspects above. That implies that the set of derivates has to be spawned for each new media element inserted into the repository. To avoid the extraordinary maintenance effort implied by this thought, we propose to create the necessary derivates *just in time*. Having the structured repository available, a device-specific derivate requested by a mobile device, which is referenced in the just received output page, is created upon this request. The newly created derivate is then stored in the repository so that it can be returned immediately upon the next request with the fitting structuring parameters. Figure 6 visualizes this workflow, which obviously causes significant performance savings by creating device-specific media once and returns it on every subsequent matching request.

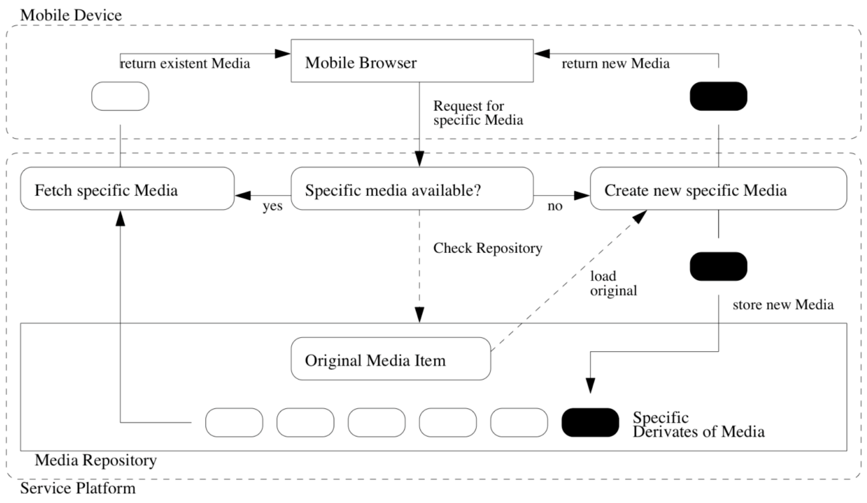

Figure 6 - Adapting media to device specifications

It is further to be reminded, that the media derivates in the repository are *not* necessarily device-specific. They are not structured by device-specificity, but by heterogeneity aspects, such as those listed above.

Multi-Channel Support

Although this paper focuses on web-based mobile services, we want to briefly discuss the possibility of communicating aside from the web-channel, too. The communication on other channels, such as SMS, Email or MMS, is implemented by modules other than the platform conceptualized in this chapter. All those modules compose the multi-channel service providing Open Dialog Platform (Comnos GmbH, 2007), as introduced in this chapter's introduction.

However, even though this chapter's platform design deals with the communication on the web-channel, it is to be clarified how *channel transitions* (from the web-channel to another and vice versa) are realized. As stated earlier, channel transitions are executed by the business logic. In the approach presented so far, we see the business logic executed during the request procession phase in the context of the page flow, when business logic states are traversed during the transition from one output state to another. This assumption implies that the business logic in question is attached to the corresponding servers controlling the specific channel communication, such as SMS centers (SMS channel) or service platforms handling multi-channel communication other than the web-channel.

A mentionable use case of the platform presented in this chapter is the applicability to extend it to enable the voice-channel. It requires the connection to a voice browser or a voice platform between our platform and the user's mobile phone. The voice platform controls the voice dialog with the user and concurrently acquires the content from our platform in form of Voice-XML, which has been created by our platform using the adaptation techniques introduced earlier. All it takes is an XSL-stylesheet capable of transforming page- and style definitions into Voice-XML documents, which are standardized for the use by voice platforms. Our page definition rules have been designed in consideration of also employing its content on the voice channel. Concluding, all additionally required components for covering the voice channel integrate perfectly into our platform, so it can be easily extended in this way.

THE PLATFORM IMPLEMENTATION

In order to validate our concept presented in this chapter, we have implemented the proposed platform, which utilizes all key functionalities described here.

The platform is an application written in Java and running on a Java servlet container. It is able to provide services to requesting mobile devices utilizing the various modular configurations. The content adaptation is working properly for devices tested with support of XHTML, WAP 1.1, WAP 1.2, HTML 4.0, CHTML (I-mode) and others. Figure 7 demonstrates an example output of a page on different mobile devices provided by the platform.

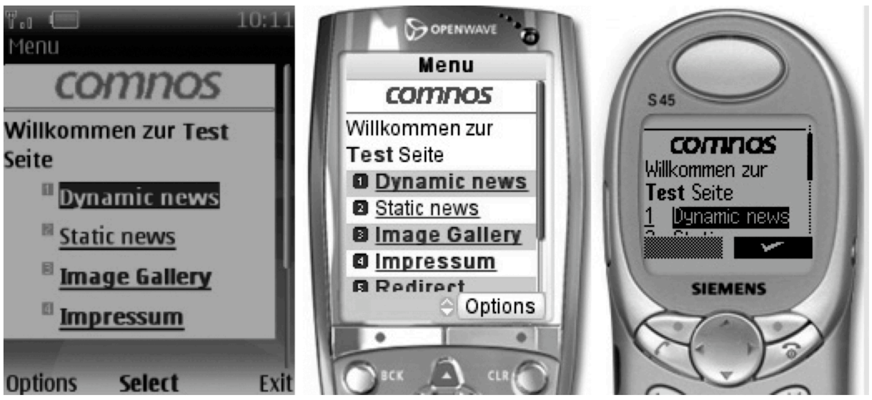

Figure 7 - Output of a generic page on 3 different devices

The service platform presented in this chapter is acting as a subcomponent of our Open Dialog Platform (Comnos GmbH, 2007). The design of the ODP is focused on multi-channel provision of mobile services employing subordinate components, each realizing communication on specified channels. The ODP runs in a J2EE environment and for that reason our mobile web-service provisioning platform integrates well into the ODP, realizing the ODP's web channel. The ODP possesses a dedicated component capable of executing business logic configurations. This business logic engine is capable of communicating with SMTP servers and SMS centers to enable multi-channel communication.

A GUI is supplementing the ODP, enabling a designer to easily compose and maintain services for mobile devices. This includes the configurations discussed in this chapter: the definition of page flows, output pages and business logic.

CONCLUSION

Summary and Outlook

The concept for adapting web content to a very heterogeneous set of mobile devices by employing a DCDB-driven adaptation mechanism has proved valid with the implementation of the platform. We are able to easily construct mobile services and provide them to any device having a web browser installed. Service construction is simple and powerful, since the most important service aspects of page flow, page definition and business logic are definable through a GUI. The service is then available to any device after the device specifications are entered into the DCDB.

Finally, the concept presented here greatly reduces the maintenance complexity implied by the heterogeneity issue, enabling mobile service providers to broaden their range of possible clients with minimal effort.

The next mid-term steps will focus on maturing the platform application, and on optimizing multi-channel communication and business logic execution.

The long-term focus lies on extending the ODP and to enable further channels other than those in a GSM network.

Related Work

Since we are addressing a current issue in this paper, several other research groups are currently engaging it, too. The most significant works, which we have identified in this research spectrum, are briefly introduced below.

Nakazawa et al. have developed a *bridging framework concept* to overcome the barrier between heterogeneous systems (Nakazawa, 2006). The system's key concept is the abstraction of semantics of the various heterogeneous domains and translating those semantics in between those domains either directly or via intermediate representations.

The Media Broker architecture, introduced by Mohdal et al. aims at interconnecting media sources and media sinks (Modahl, 2004). Since both sources and sinks are settled in their respective, highly heterogeneous domains, this research group faces similar issues as presented in this chapter. They, however, address it by abstracting a set of possible data types and employing compatibility checks when attaching media sources to media sinks.

The research group Chan et al. aims at circumventing parts of the device heterogeneity issue by exploiting the widely spread device-capability of J2ME support (Chan, 2005). Their Gaia Microserver enables their Gaia Ubiquitous computing platform to be utilized on mobile devices. A generic J2ME client is used to select and install a platform-dependent distribution of the microserver on the device. Since the selection procedure is transparent, this approach may be regarded as platform-independent, hence decoupling the microserver's execution from certain device heterogeneity aspects.

Acknowledgements

This project has been conducted by the Technische Universität München (Technical University of Munich) and the Comnos GmbH, funded by the Bavarian ministry of economy (Bayerisches Staatsministerium für Wirtschaft, Infrastruktur, Verkehr und Technologie) under grant IUK219/002.

REFERENCES

Apache Sofware Foundation (2007). J2EE frameworks by the Apache Software Foundation, cited Novemver 2nd, 2007, from http://apache.org

Comnos GmbH (2007). Open dialog platform, cited November 2nd, 2007, information available at http://www.comnos.de/?menu=so&content=page_so_platform.htm

HTTP (2007). Hypertext transfer protocol, cited November 2nd, 2007, specification available at http://www.w3.org/Protocols

Open Symphony (2007). Webwork application framework, cited November 2nd, 2007, available at http://www.opensymphony.com/webwork

Openwave System Inc. (2007). Openwave mobile browser, cited November 2nd, 2007, information available at http://www.openwave.com

Spring (2007). Spring framework and spring web flow, cited November 2nd, 2007, available at http://www.springframework.org

Sun Microsystems Inc. (2007). Java server faces, cited November 2nd, 2007, available at http://java.sun.com/javaee/javaserverfaces

WBMP (2007). Wireless application protocol bitmap format, cited November 2nd, 2007, specification available at http://www.openmobilealliance.org

XSL-T (2007). XSL transformations, cited November 2nd, 2007, specification available at http://www.w3.org/TR/xslt

Ellick Chan, Jim Bresler, Jalal Al-Muhtadi & Roy Campbell (2005). Gaia microserver: An extendable mobile middleware platform, In *Third IEEE International Conference on Pervasive Computing and Communications (PerCom'05)* (pp. 309-313), IEEE Computer Society

Ricardo Couto A. da Rocha & Markus Endler (2005). Evolutionary and efficient context management in heterogeneous environments, In *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, (pp. 1-7), New York, NY: ACM Press

Martin Modahl, Ilya Bagrak, Matthew Wolenetz, Phillip Hutto & Umakishore Ramachandran (2004). Mediabroker: An architecture for pervasive computing, In *Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)* (p. 253), IEEE Computer Society

Jin Nakazawa, H. Tokuda, W.K. Edwards & U. Ramachandran (2006). A bridging framework for universal interoperability in pervasive systems, In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006)* (p. 3), IEEE Computer Society

Robert Schmohl & Uwe Baumgarten (2007). Mobile services based on client-server or p2p architectures facing issues of context-awareness and heterogeneous environments, In *PDPTA '07: Proceedings of the 2007 international conference on parallel and distributed processing techniques and applications* (pp. 578-584), CSREA Press

Roy Want &Trevor Pering (2005). System challenges for ubiquitous & pervasive computing, In *ICSE '05: Proceedings of the 27th international conference on Software engineering* (pp. 9-14), New York, NY: ACM Press

KEY TERMS

Mobile services are services utilized by a user of a mobile device. Although they are not necessarily restricted to the mobile sector, their focus clearly lies on the utilization on mobile devices. Mobile services are facing significant constraints concerning their expressional power. Their target devices are highly heterogeneous, have limited output capabilities and constrained hardware capabilities concerning computational power, energy and communication.

Content adaptation is a necessary process to enable mobile devices to display generic content properly. The content is adapted to a mobile device's display capabilities. This is necessary, since those capabilities are very heterogeneous among mobile devices.

Markup languages are expressional tools enabling to define structured content. They use tags and attributes to both structure data and enrich it with auxiliary information.

Page flow defines a web application's sequence of pages, which is traversed by its user. It normally distinguishes between states for outputting data and states for executing business logic.

Request-response-cycle is the time period between a user issuing a request and the web application returning the appropriate page to the user.

Business logic is the definition of any logic in a web application. It is usually executed during a request-response-cycle.

XML-validity denotes the conformity of an XML-document to its corresponding scheme.

XSL-T is a technique for transforming XML-documents. The source is usually corresponding to an XML-schema, so that an XSL-stylesheet can define all the transformation rules for the corresponding sources.