Content Adaptation for Heterogeneous Mobile Devices using web-based Mobile Services

Robert Schmohl [1], *Uwe Baumgarten* [2], *Lars Köthner* [3]

Abstract:
Recent advances in mobile computing have spawned a very heterogeneous environment of mobile devices. Those devices have different capabilities in providing mobile services to the user, implying the challenge of considering heterogeneous devices during mobile service development. This especially encompasses the task of adapting the content, which a mobile service provides to a specific mobile device. In this paper we present an approach using a service platform, which utilizes 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.

1 Introduction

Since mobile computing is getting increasingly popular, the development of mobile services is getting increasingly complex implying new challenges to be handled [14] [15]. One of those challenges is the highly heterogeneous environment of mobile devices. 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 market. However, although this approach may work in practice, it is a temporary solution of the problem, requiring a high level of maintenance implied by the rapid evolution of technologies.

Mobile devices' heterogeneity can be divided into hardware and software heterogeneity [11]. 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, the 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.

[1]Technische Universität München, Institut für Informatik, 85748 Garching, Germany, schmohl@in.tum.de
[2]Technische Universität München, Institut für Informatik, 85748 Garching, Germany, baumgaru@in.tum.de
[3]Comnos GmbH, 80335 München, Germany, lars.koethner@comnos.de

- *Device output capabilities:* Devices have very different capabilities in processing output visually and acoustically (see following section 4).
- *Logical communication 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, that 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 the service. 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). The latter aspect omits the need of creating services by means of programming (see following section 3.3).

The concept has been developed in a joint effort by the Technical University of Munich and the Comnos GmbH, to complement an existent mobile service platform, the "Open Dialog Platform (ODP)" [2], capable of providing services by SMS, MMS and email. The conceptual design introduced here aims at complementing the ODP by enabling service provision on the web channel.

The rest of this paper is structured as follows: In section 2 we discuss the aspects relevant for a web-platform developed for mobile service provision. Section 3 presents an overview about the utilized concept and the corresponding platform design. Section 4 discusses the approach to solve the heterogeneity problem. In section 5 we briefly present our implementation of the platform, and we subsequently conclude this paper in section 6.

2 Aspects of a Web-Framework

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 [2], which runs in a J2EE environment and which is intended to host our mobile service platform implementation (see section 5). The evaluation has encompassed frameworks developed by the Apache Foundation [1], such as Struts, Turbine, Tapestry, Velocity and Cocoon, as well as Spring [6], WebWork [4] and Java Server Faces [7]. 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 and page flow control:* Those aspects describe how incoming client requests are handled. This includes the issue of assigning existent sessions to its users and properly controlling their page flow. A *page flow definition* is defined by the sequence of pages, which a user can request in a web application. XML is a reasonable format to represent both definitions of request mapping and page flow.

- *Output rendering:* Since we are dealing with a highly heterogeneous set of possible clients, the device-specific adaptation of output, that is eventually returned to the requesting client, 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.

- *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, This makes XML a suitable format for the representation of configurations.

- *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.

3 Platform Design

The aspects from section 2 represent a frame for the development of our concept of the web platform presented in this paper. They can be interpreted as basic requirements, which need to be met in the design discussed in this section.

3.1 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*.

Device detection An incoming request from a mobile device is received by the platform via the HTTP protocol [3]. 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 states either trigger the output of web pages to the client or the execution of business logic. Put in sequence, those states create a flow, which basically denotes the workflow of a service.

Intermediary page generation The output information is primarily generated as a device-independent meta-page, solely including static content information. It is generated from a static page definition defined by the service, and it is called *intermediary*, since it is not yet final due to lacking device-specifity. Its generation consists of including all dynamic data, which is available not until runtime. This may include data defined by the execution of business logic, data passed by the user in the request, etc. The resultant intermediary page includes all of the content information to be sent to the client device. Both page definition and intermediary page are valid XML documents.

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 [9], a common XML transformation technique. The adaptation process includes the conformance to the device's support of both markup language and media (see section 4). 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. The resultant output page is subsequently returned to the client, completely tailored to its device capabilities. Figure 1 visualizes the workflow discussed above.

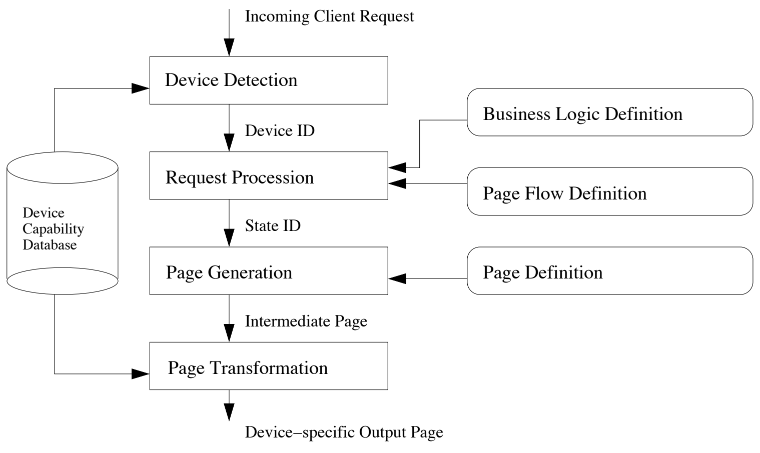

Figure 1: Workflow

3.2 Architectural Draft

The architectural draft presented here strongly reflects the workflow described in section 3.1. The platform is composed of 3 basic components: The *device detector* component handles the device detection and the connection to the DCDB. It is responsible for loading device data into the platform. The *request processor* handles web-specific procession of incoming client requests, such as request mapping and page flow procession. Finally, the *output generator* controls the generation of the intermediary page and its adaptation to the client device.

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 2 visualizes this proposed architecture.

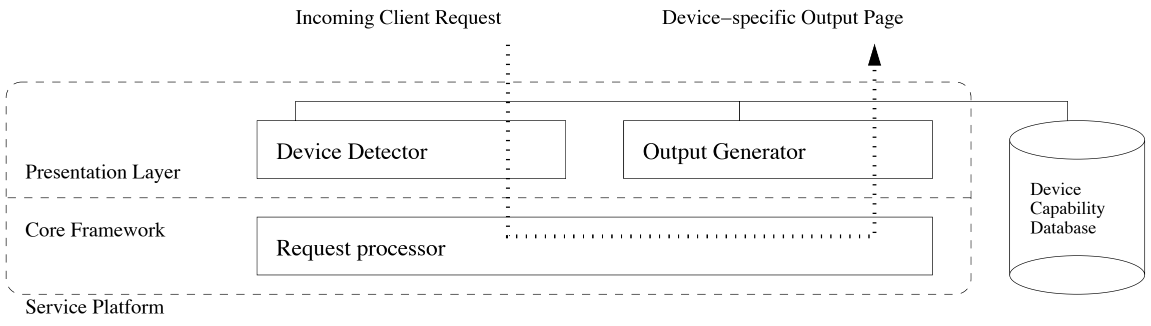

Figure 2: Architectural Draft

3.3 Configuration

Our platform proposal utilizes 3 configuration entities. The *page flow definition* defines the page flow as discussed in section 3.1, the *business logic definition* defines the business logic executed during page flow traversing, and the *page definition* defines the structure and content of an output page as discussed in section 3.1. 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 1). 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. Represented in a format readable both by men and machines, such as XML, the effort to create services is minimized.

4 Content adaptation

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.

4.1 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 or an external product, such as the externally licensed OpenWave Browser [5]. Hence, markup support is not dependent on devices, but on the mobile devices' browsers. This heterogeneity aspect is subject of the subsequent section 4.2.

- *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, an newer mobile device may support colored Jpeg images whereas an older device may only be able to display WBMP [8] 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. This heterogeneity aspect is subject of the subsequent section 4.3.

- *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 focusses on the web channel, we state this aspect on account of completeness and we briefly discuss it in section 4.4.

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.

4.2 XSL-T Transformation and Parameterization

As outlined in section 3.1, 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, determined during the device detection phase, is used to parameterize the transformation process. That can be decomposed into the following steps:

1. *Stylesheet selection:* XSL-T transformations are defined by stylesheets. 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.

4.3 Adaptation of Media Content

In section 4.1 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 section 3.1) and it is passed in the form of parameters to the transformation engine prior to the transformation process (see section 4.2). Then, the transformed output page does not reference the original media specified in the page definition, but the 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 3 visualizes this approach.

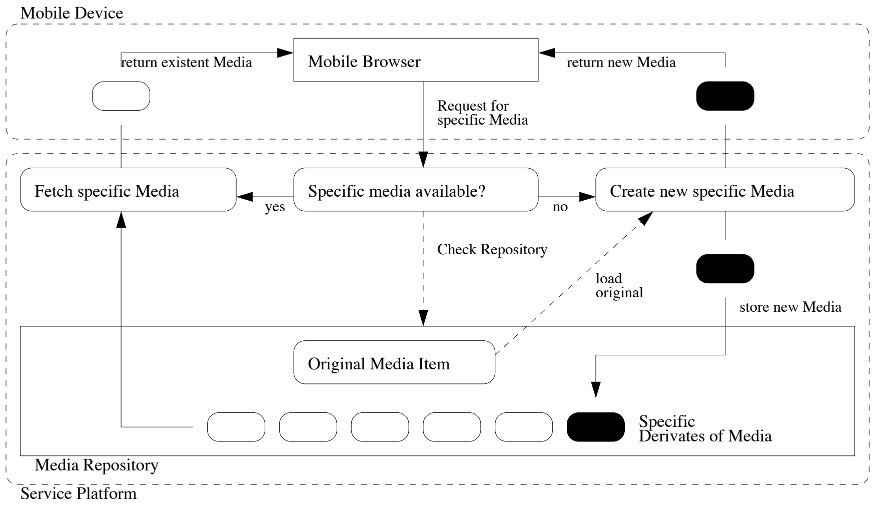

Figure 3: Adapting Media to Device

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.

4.4 Multi-channel Support

Although this paper focusses on web-based mobile services, we want to briefly discuss the possibility of communicating aside from the web-channel, too. In the approach presented so far, we see the business logic executed during the request procession phase (see section 3.1). This implies that the business logic in question is attached to the corresponding servers controlling the specific channel communication, such as SMS centers (SMS channel) and voice platforms (voice channel), or even service platforms handling multi-channel communication other than the web-channel (as described in the following section 5).

5 The Platform Implementation

In order to validate our concept presented in this paper, we have implemented the proposed platform, that 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 4 demonstrates an example output of a page on different mobile devices provided by the platform.

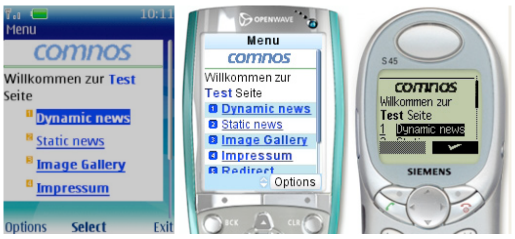

Figure 4: Output of a generic Page on three different Devices

The service platform presented in this paper is acting as a subcomponent of our Open Dialog Platform [2]. The design of the ODP is focussed 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, voice platforms 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 paper: the definition of page flows, output pages and business logic.

6 Conclusion

6.1 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 and page definition 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, on optimizing multi-channel communication and business logic execution, and on finishing integrating it into the ODP. After completion of this issues we will have the ODP enabled to provide services for mobile devices on any channel in a GSM network. The long-term focus lies on extending the ODP and to enable further channels other than those in a GSM network.

6.2 Related Work

Since we are addressing a current issue in this paper, several other research groups are currently engaging it, too. This accounts for the uMiddle framework [13], Media Broker [12] and the Gaia Microserver [10], to name a few.

References

[1] Apache software foundation. http://apache.org.

[2] Comnos gmbh, open dialog platform. http://www.comnos.de.

[3] Http - hypertext transfer protocol. http://www.w3.org/Protocols/.

[4] Open symphony, webwork application framework. http://www.opensymphony.com/webwork.

[5] Openwave system inc., openwave mobile browser. http://www.openwave.com.

[6] Spring framework and spring web flow. http://www.springframework.org.

[7] Sun microsystems inc., java server faces. http://java.sun.com/javaee/javaserverfaces.

[8] Wireless application protocol bitmap format. http://www.openmobilealliance.org.

[9] Xsl transformations (xslt). http://www.w3.org/TR/xslt.

[10] Ellick Chan, Jim Bresler, Jalal Al-Muhtadi, and Roy Campbell. Gaia microserver: An extendable mobile middleware platform. *percom*, 00:309–313, 2005.

[11] Ricardo Couto A. da Rocha and Markus Endler. 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*, pages 1–7, New York, NY, USA, 2005. ACM Press.

[12] Martin Modahl, Ilya Bagrak, Matthew Wolenetz, Phillip Hutto, and Umakishore Ramachandran. Mediabroker: An architecture for pervasive computing. *percom*, 00:253, 2004.

[13] Jin Nakazawa, H. Tokuda, W.K. Edwards, and U. Ramachandran. A bridging framework for universal interoperability in pervasive systems. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 3–3, 2006.

[14] Robert Schmohl and Uwe Baumgarten. 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*, pages 578–584. CSREA Press, 2007.

[15] Roy Want and Trevor Pering. System challenges for ubiquitous & pervasive computing. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 9–14, 2005.