AN ADAPTABLE MIDDLEWARE FOR PERSONALIZING WEB APPLICATIONS

Zahi Jarir [1,2] And Mohammed Erradi [2]
[1] *Cadi Ayyad University, Faculty of Sciences Semlalia Marrakech*
Department of Computer Science
B.P. 2390, Bvd. Prince My Abdellah, Marrakech, Morocco

[2] *Computer Networks and Multimedia Research Group , UFR-RT*
Mohamed V Souissi University, ENSIAS
B.P. 713, Agdal, Rabat, Morocco.

Keywords: Web Personalization, Dynamic Reconfiguration, Computational Reflection, Reflective Middleware, Separation of concerns, Component-based Application, EJB, JOnAS, Web application.

Abstract: The personalization is an important topic for the Web industry. It consists in providing the capabilities to accommodate Web applications to user's requirements such as defining preferences on the execution of the application, associating the provided application to a specific terminal, specifying or modifying QoS parameters, and so on. The contribution of this paper is to present a solution to ensure an advanced Web application personalization by focusing on the middleware level rather than the application level. We provide an enhanced architecture to personalize Web applications using the EJB technology. An implementation using JOnAS environment is presented. It has the advantage to adapt and/or reconfigure Web application's behavior at runtime according to the user's specific needs.

1 INTRODUCTION

Web applications users become more and more experienced and thereafter require more advanced and sophisticated Web applications. In fact these "smart users" are interested in Web applications that are easy to reconfigure and/or adapt to their new requirements. The users may require defining preferences related to the application's execution such as fluctuating network bandwidth, and/or associating the Web application to a specific terminal, and so on. For instance certain users may wish to adapt their Video-On-Demand web application on their PDA instead of their laptop by selecting voice rather than sub-titles. The capability to accommodate applications to the user's preferences such as technological restrictions, mobility requirements, resources constraints, etc. is called personalization.

The personalization may consist in offering the possibility to adapt dynamically or statically Web applications to build ad-hoc applications related to the user's needs. In the new provision environments, Web applications Providers are looking to introduce new architectures and techniques to allow users to have an active role and to be involved in the personalization process of their applications.

Personalizing Web applications is therefore a challenging task and becomes even more powerful when applied to advanced and multimedia applications. Consequently, new tools and architectures for personalization, especially dynamic personalization, are required. In this direction a new paradigm, called engineering adaptive Web applications, have been emerged.

The rest of this paper is organized as follow. In Section 2, we discuss related work. In Section 3, we argue the motivations to choose EJB platform and thereafter we expose the main entities of the EJB architecture. In Section 4, we show how the EJB architecture can be extended to allow dynamic adaptability of services. In Section 5, we apply the suggested approach to personalize a Web application which consists in interacting with a Multimedia Conferencing System before concluding in Section6.

Jarir Z. and Mohammed Erradi A. (2005).
AN ADAPTABLE MIDDLEWARE FOR PERSONALIZING WEB APPLICATIONS.
In *Proceedings of the First International Conference on Web Information Systems and Technologies*, pages 287-292
DOI: 10.5220/0001234602870292

2 RELATED WORK

Several approaches in adaptive Web applications are currently investigated, using different techniques and methodologies, in different areas e.g. eCommerce, Distributed eLearning Environments, Telecommunication Services, etc. These approaches concern mainly the following topics:

a. *Web Information Personalization* that focuses on searching and browsing the Web information using personalized Web search systems. Recently researchers are interesting to define how to enable personalization functionality to personalize the interaction with web content (Henze et al., 2004; Shahabi, 2003).

b. *Web Presentation Personalization* that contributes to tailor the information presented and the structure to the user's preferences, knowledge or interests. This topic focuses on how adaptation can be implemented through the manipulation of links, and content and presentation of nodes (Fortier et al., 2001; Goderis et al., 2001; Koch et al., 2002). One sub-category of this context is terminal adaptivity (Hinz et al., 2004).

c. *Web Application Personalization*, that contributes to tailor the Web application structure and/or behavior to satisfy specific user requirements. To address this kind of personalization there is mainly two ways :

- Using conceptual approaches that consist in modeling and designing personalized Web applications. The aim of this approach is to bring dynamic personalization support in Web conceptual modeling constructs by creating new conceptual modeling approach or extending the existing ones e.g. OOH, OOHDM, etc. (Garrigos et al., 2003).
- Using implementation approaches that consist in modifying the implementation details of the developed Web applications. These approaches introduce personalization features using different techniques such as mobile agent, reflection techniques, and so on (Maknavicius et al., 1999; Jarir et al., 2002).

Most of the evoked researches focus mainly to a specific Web application concern such as building tailored presentation interface, showing personalized information, adapting application's functionalities, etc. However less attention has been paid to build an architecture that covers almost all the concerns of Web personalization.

Building an architecture for personalizing Web application imply to deal with different personalization concerns that must be seamlessly integrated. That's why we have elaborated as a first step a deep taxonomy showing the varieties of personalization concerns. Based on this taxonomy, a new problem emerges: what is the best level among the application level and middleware level to deal with the diversity of personalization concerns (diversity of users devices, evolvable execution context, etc.)?

Recall that Web applications are usually developed over a Middleware which in turn is implemented over a protocol stack to ensure data communication and to manage the communication resources. Performing dynamic changes to Web applications become a tedious task and need to consider changes at the underlying levels especially at the middleware level. The idea behind this approach is that applications built on the top of the middleware rely on it for all interactions with the execution environment, so adapting the middleware allows us to indirectly adapt the applications. Moreover at the middleware level, the Web application may be adapted to a given configuration, either by replacing some components of code or by adding new components in order to enhance existing functionalities. Furthermore it is very difficult to adapt Web application to the continuous variations of their execution contexts at the application level.

3 WHY ENTERPRISE JAVABEANS ARCHITECTURE

Object-Oriented Middleware technologies such as CORBA or Java RMI have proved their suitability for standard client-server applications. However, such platforms do not provide the required levels of adaptation and/or reconfiguration that are needed to accommodate the diversity of modern distributed applications including support for multimedia, real time, mobility, etc. This motivates many middleware research groups to built new and advanced middleware technologies.

Sun Microsystem, OMG and Microsoft are aware of these limitations that's why the current developments of the Enterprise Java Beans (EJB), CORBA Components Model and COM+ are proposed. The main focus of these platforms is to alleviate the application-level programmability issues by hiding, within the so-called component containers, a large part of the complexity with more declarative interfaces.

The growing popularity of EJB architecture is due to the advantages offered to the distributed and Web-based applications, e.g. faster application development, ability to build complex applications, separation of business logic from presentation logic,

application interoperability, etc. Furthermore, currently there are more than 30 implementations (free and commercial) of EJB servers and that number is increasing (Emmerich et al., 2002). Therefore, we choose the EJB technology as an example of component-based middleware to prove that Web application personalization can be dealt with at the middleware layer.

Enterprise JavaBeans (EJB) is a specification and architecture for the development and deployment of distributed server-side, transactional, and secure business application components. Therefore EJB servers reduce the complexity of developing middleware by providing automatic support for middleware services such as transactions, security, database connectivity, and more. The EJB specification defines also several standard roles and responsibilities that introduce three fundamental. These entities are EJB server, EJB Container and EJB component.

However, the EJB platform does not address the needs for adaptation and extension required in several applications. Our suggested solution focuses on the EJB architecture to show how it can be extended to allow reconfigurations and extension at runtime providing an adaptable EJB infrastructure (Jarir et al., 2002). This research was supported by the RNTL project ARCAD (ARCAD, 2005).

4 TOWARDS AN ADAPTABLE EJB INFRASTRUCTURE

In this section we first show how to introduce runtime adaptability in the EJB environment. Then we present our approach for an adaptable EJB infrastructure. This infrastructure is made using the computational reflection (Maes, 1987), which is a suitable technique for dynamic adaptation features.

4.1 How to make EJB adaptable?

According to the separation of concerns paradigm used by the EJB architecture, the code of an EJB application is split in two parts: the functional code, representing EJB components, and the non-functional code, representing middleware services (e.g. transactions, persistence, security, etc.). However, the configuration between EJB components and Middleware services is only supported at deployment-time using a declarative deployment descriptor. This descriptor, presented in XML format, defines a set of accessor methods for setting and getting information about the Entreprise Java Beans being deployed.

To make EJB adaptable we need to make explicit the separation between functional code and non-functional code. Then, the associations between these two kinds of codes can be modified at runtime according to the user's requirements, to personalize its application. This reconfiguration is made by identifying which piece of the functional code is affected (and how it is affected) by the non-functional code.

To ensure an advanced adaptability of an EJB application to the desired changes, we need to focus on the EJB container layer. This is because the EJB container is an intermediary between the EJB components and the outside world especially between the EJB component and the access to various resources and EJB services. The containers are generated statically using the information provided by the EJB component's deployment descriptors. These descriptors cannot modify the associations between the EJB components and the EJB services at run-time without modifying the containers.

4.2 Computational reflection

Computational reflection, or just reflection, has been proposed as a solution to the problem of creating applications able to maintain, use, and change representation of their own designs. It is defined as the capability to monitor and modify dynamically the structure and the behavior of a system. A reflective system is therefore able to use self-representations to extend, modify and analyze its own computation (cf. Figure 1).

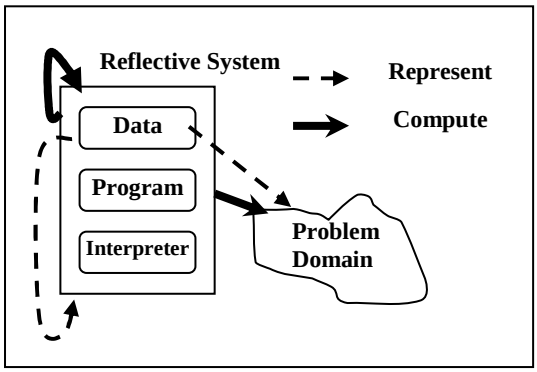

Figure 1: Reflective system

4.3 The adaptable EJB infrastructure

JOnAS (Java Open Application Server) (Jonas, 2005) is an Open Source implementation of the J2EE™ specification. It is a pure Java™ implementation of this specification that relies on the JDK. It is part of the ObjectWeb Open Source initiative. The Opening of JOnAS environment opens us the way for introducing the reflection features to try to make EJB architecture adaptable. In addition, JonAS environment offers an open source tool, called the GenIC (Generate Interposition Classes) that allows generating the EJB container code. Therefore this tool will guarantee to set up our approach by focusing on the EJB container layer as mentioned earlier.

In order to respect the EJB container specification, we have delegated the task of a dynamic composition of services to another object called DynamicComposite, representing the meta-object of the EJB container as shown in Figure 2. The set up of this indirection is made thanks to the computational reflective features that allow diverting all methods call from EJB container to its associated DynamicComposite object. This object is able to compose dynamically attached or detached EJB services before or after sending the method call to the EJB component. Therefore the DynamicComposite object will be responsible for playing the role of a dynamic composer of the EJB services.

To perform the required reconfiguration (attachment and/or detachment of EJB services) for each application, a generic adaptation engine is introduced. This reconfiguration is guided by the adaptation policies, described in XML format, and they are initiated by the engine when significant evolutions of the environment are detected by a simple monitoring framework, consisting of a collection of probes (CPU usage, battery life, bandwidth measure…).

These adaptation policies are:

- *System policies*, consisting in sets of rules of the form *condition* ⇒ *action*, where the condition is related to the execution environment (as reified by the monitoring framework), and the action is either the attachment or detachment of a specific services, possibly with configuration parameters.
- *Application policies*, which define groups of EJB components according to their runtime properties and bind existing system policies to these groups.

The following XML code presents an example of a system policy named “bandwidth-policy”, which are interpreted at run-time by the adaptation engine.

```
<system-policy name="bandwidth-policy">
 <rule>
 <when>
  <less-than>
   <property-value name="/system/network.bandwidth"/>
   <number value="40000"/>
  </less-than>
 </when>
 <ensure>
  <detached service="VideoService"/>
  <updated service=”AudioService”>
  <parameter name=”SoundEncoder” property-value=”classLpc”/>
  </updated>
 </ensure>
 </rule>
</system-policy>
```

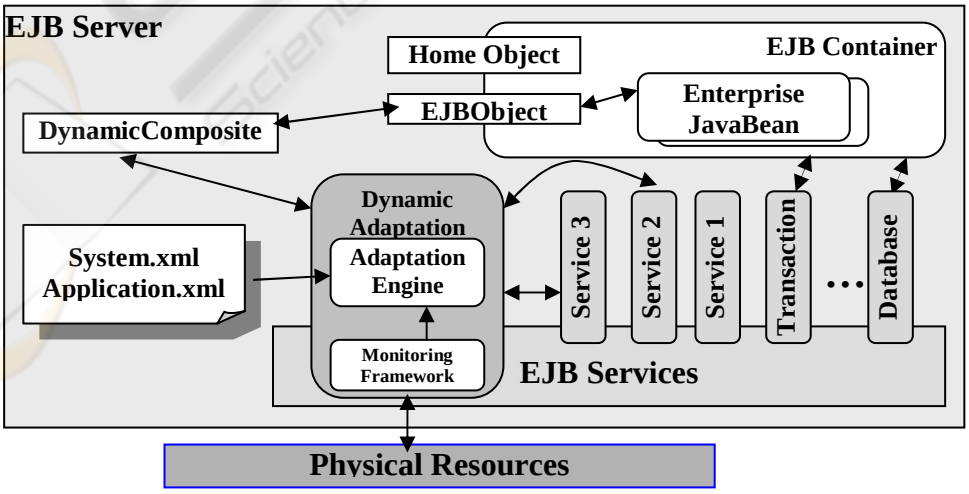

Figure 2: An adaptable EJB infrastructure

5 AN ENHANCED EJB ARCHITECTURE FOR WEB APPLICATIONS PERSONALIZATION

The increasing popularity of the World Wide Web and the diversity of hardware appliances make the Internet users crave to have the privilege to control the functionalities of the Web applications to which they are subscribed. For example a user may need to obtain a MPEG2 Format for a movie, obtained from a Video-on-Demand server, instead of a classic VHS quality received by default.

The Web application personalization considers many types of reconfiguration. These reconfigurations may concern changes of:

- *Parameters of the application* that concerns the application's data, as for example QoS parameters of the multimedia applications (e.g. debit, video resolution, etc.)

- *Functional aspect of the application* that concerns the application's behavior such as:
  - *Adaptability* that affects the application's behavior without calling new components or functionalities within the application. This personalization consists in activating and/or deactivating some of their already existing functionalities.
  - *Extensibility* that corresponds to introduce new additional behavior or functionalities in the application to answer a specific need.
- *Technological aspect* that concerns the modification of the application in order to be executed on different platforms (e.g. operating system, etc.), different types of terminals, the run-time variations of availability of certain resources such as CPU, memory, communication capacities, and so on.

To build personalizable Web application, a more flexible architecture, that takes into account those varieties of adaptation and/or reconfiguration, is needed.

To build a more flexible architecture for Web applications personalization, we have reused the adaptable JOnAS EJB infrastructure that allows component-based applications to be aware of, and adapt to, the variations in the execution context. Therefore our goal was to see how this infrastructure could be transformed in response to all varieties of reconfiguration and/or adaptation studied before.

Figure 3 shows the elaborated architecture for Web applications personalization. This EJB-based architecture has the advantage to be able to ensure a runtime tailorability of the Web applications behavior. This tailorability concerns both the behaviors of each EJB service and also each other's composition that build the desired Web application. More precisely, the dynamic personalization takes into account the attachment and/or detachment of a service and also the update of a specific attached service behavior. This personalization can be activated according to:

- The expressed user's requirements on the Graphical User Interface (GUI). These requirements are thereafter translated to a specialized policy, called personalization policy, that will be performed by the adaptation engine incorporated inside the Dynamic Adaptation service. After the required actions expressed by this policy are handled by the DynamicAdaption Service, the EJB Client will thereafter modify the GUI presentation according to these changes. The following code shows an example of a personalization policy:

```
<personalization-policy>
 <updated service name="VideoService">
   <parameters name="resolution" value
="1024x768"/>
   <parameters name="VideoEncoder" value
="MPEG1"/>
 </updated>
```

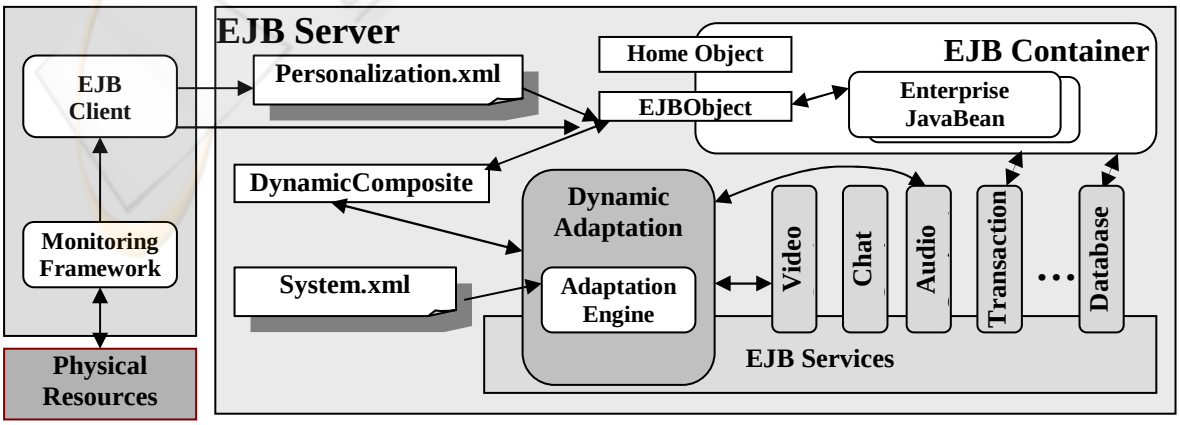

Figure 3: An EJB architecture for Web applications personalization

```
  <detached service name=”chat”/>
  <attached service name="AudioService">
    <parameters name=”AudioEncoder” value
="classGSM"/>
  </attached>
</personalization-policy>
```

- The variations of the client environment that considers the technological restrictions, the continuous variations of the execution environment, etc. This detection is handled by the monitoring framework, placed at the client-side, in response to the conditions introduced by the system policies rules. To allow the EJB client to be informed about what type of variations can be taken into account to notify the EJBObject, we have injected implicitly the conditions of the system policies rules during the initialization of the application.

6 CONCLUSION AND PERSPECTIVES

The contribution of this paper deals with Web applications personalization. Specifically, we have shown how the personalization can be dealt at the Middleware level by developing an adaptable JOnAS EJB infrastructure. This infrastructure consists in modifying at run-time and with a fine granularity the association between EJB components and middleware services. Based on this EJB infrastructure, we have presented an adaptable EJB architecture to personalize Web applications. This enhanced architecture allows users to take an active role in personalizing dynamically their Web applications, and also enables Web applications to be aware of, and to adapt to, the variations in the execution environment.

Currently we are interested in the extensibility of the Web application by adding a required service from other application service provider. However this extensibility may cause deviation from desired behavior or systems failure. This issue generates a broad topic, named feature interaction problem that requires service (or feature) interference detection and resolution.

REFERENCES

ARCAD, http://arcad.essi.fr

Emmerich, W., & Kaveh, N., 2002. Component Technologies: Java Beans, COM, CORBA, RMI, EJB and the CORBA Component Model. Proc. of the 24th Int. Conference on Software Engineering, Orlando, Florida. pp. 691-692. ACM Press.

Fortier, A., Rossi, G., & Cappi, J., 2001. Using Meta-Level Techniques to Personalize O-O Applications”, Object-Oriented Programming, Systems, Languages, and Applications OOPSLA 2001/ECOOSE.

Garrigos, I., Gomez, J., & Cachero, C., 2003. Modelling Dynamic Personalization in Web Applications. In Proc. 3 rd Int. Conf. Web Engineering, volume 2722 of LNCS., pages 472–475. Springer Verlag,

Goderis, S., Rossi, G., Fortier, A., Cappi, & J., Schwabe D., 2001. Combining Meta-level and Logic-Based Constructs in Web Personalization, In Proc.of the 6th International Computer Science Conference on Active Media Technology, Pages: 57 – 64.

Henze, N., & Kriesell, M., 2004. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementations. First International Workshop on Engineering the Adaptive Web.

Hinz, M., Fiala, Z., Wehner, F., 2004. Personalization-based Optimization of Web Interfaces for Mobile Devices; In: Proceedings of 6th International Conference on Human Computer Interaction with Mobile Devices and Services, Glasgow, Scotland.

Jarir, Z ., David, P.C. & Ledoux, T., 2002. Dynamic Adaptability of services in Entreprise JavaBeans Architecture , Seventh International Workshop on Component-Oriented Programming /ECOOP 2002, Malaga, Spain.

Jarir, Z., & Erradi, M., 2002. Telecommunication Services Customisation , Association for the Advancement of Modelling and Simulation Techniques in Enterprises, Vol. 23 n°3, pp. 1-14, AMSE-Journal.

JOnAS, http://www.evidian.com/jonas

Koch, N. et al, 2002. Patterns for Adaptive Web Applications. In 7th European Conference on Pattern Languages of Programs, Irsee, Germany.

Maes, P., 1987. Concepts and Experiments in Computational Reflection , in Proc. Conference on Object-Oriented Programming systems, Languages and Applications, Orlando, FA, pp. 147-155.

Maknavicius, L., Koscielny, G. & Znaty, S., 1999. Customizing Telecommunication Services : Patterns, Issues and Models, , vol. 1597, pp. 194-209, Lectures Notes in Computer Science Springer-Verlag.

Shahabi, C., 2003. Web Information Personalization: Challenges and Approaches, In the 3nd International Workshop on Databases in Networked Information Systems , Aizu-Wakamatsu, Japan.