A Meta-Level Architecture for Adaptive Applications

Fabrício J. Barth, Edson S. Gomi
Laboratory of Knowledge Engineering (KNOMA)
Polytechnic School. The University of São Paulo, Brazil
E-mail: {fabricio.barth, edson.gomi}@poli.usp.br

Abstract

The goal of this work is to investigate meta-level architectures for adaptive systems. The main application area is the user modeling for mobile and digital television systems. The results of a set of experiments performed on the proposed architecture showed that it is possible to reuse the components responsible for user modeling if they are designed as meta-level components.

1 Introduction

Currently most applications are developed for a large variety of users. An important issue is how to create adaptive applications that enable systems to satisfy heterogeneous needs. An adaptive system is a system with the ability of adapting its own behavior (with respect to performance and functionalities) to individual users needs. Some examples of system functions that can be adapted are: information retrieval, product recommendation, learning support, and user interfaces. Beyond the set of basic functions, systems may implement mechanisms that permit automatic development of a user profile, representing it in a user model. A user model is a knowledge representation of the preferences which determine the user's behavior. Preferences are the basic information necessary to provide system adaptability.

In traditional approaches, user modeling components have been created integrated with other system elements, without a specific component responsible for it. With the growing complexity of the adaptive systems, developers need new methodologies, concepts and techniques to overcome design difficulties and simplify the development process.

In this work we propose a meta-level architecture to separate the components responsible for the acquisition and manipulation of user modeling from other components of the adaptive system. This architecture has base-objects in its lower level and meta-objects in its upper level. Meta-objects can be defined as objects that describe the user model. The meta-objects can control a base-object, in order to modify its structure and behavior at execution time. This type of control is possible due to the reflective computing concept. Reflective computing is defined as the activities that a computational system execute over itself, in a different way from other processes, to solve their own problems and to search information about their own processes in real time.

To validate these architectures, experiments with interface adaptive systems will be conducted. The adaptation capacity, the reusability degree and the reduction in complexity of the implemented systems will be measured.

This paper is organized as follows. In section 2 the concepts of adaptive systems and user modeling are presented; in section 3 the concepts of meta-level architecture and reflective computing are described; in section 4 we propose an architecture that joins the meta-level concepts and the adaptive systems; in section 5 the implementation of a mobile system with adaptive interface is described. This system uses the architecture proposed in this paper. Finally, in section 6, the final considerations are presented.

2 Adaptive Systems

An adaptive system can be defined as a system with the ability to adapt its own behavior (i.e. functionalities) to the individual needs, expectations and abilities of the users [1]. This kind of system should be able to implement mechanisms that allow the creation and storage of the user model.

A user model is composed of the knowledge of the individual preferences that determine the user's behavior. Preferences are composed of information that is directly necessary to the adaptation of the system's behavior to the user's interests [2, 3]. For example, how many times, how frequent and for how long a user accesses one link in a web page.

Moreover, the user model may have personal information about the user such as age and profession. This information is not directly necessary for adapting a system to the user, but can be utilized to create user stereotypes. This allows the system to anticipate some of the user's behaviors [4, 5, 6]. Systems with that type of antecipation has shown benifits to both users and service providers [7]. However the development and maintenance of those personalizable systems are very complex and expensive, which suggests the need of a new approach for such undertaking [8].

In the majority of the related research works, user modeling is performed by the application system, and often there is no clear distinction between the system components that are used for user modeling purposes and the components that perform others tasks [8, 9].

In this context, we believe that the utilization of a meta-level architecture will help in the development of adaptive systems.

3 Meta-Level Architecture

Computational reflection is the activity executed by a computational system when the system computes (and possibly affects) its own computations. Reflection is defined as a way of introspection, in which the system tries to draw conclusions about its computations and eventually influence them [10].

Computational reflection defines a new software architecture paradigm. This architecture model is composed of a base-level, where we find the system objects that implement the system functionalities, and the meta-level, where we find the data structures and actions to be performed over the base-level objects [11]. The domain limits is the most interesting aspect of the reflexive architecture, not only because it permits the constructions of adaptive systems but also because it stimulates the reuse of components. The main point is to allow the application programmer to concentrate on the solution of a specific problem of the application domain. In this aspect, the meta-level architectures have been adopted to express non functional characteristics of the system, such as reliability and security, in an independent way from the application domain.

The basic concepts of computational reflection are [12]:

1. to separate the basic facilities from the non basic facilities through architectural levels;
2. the basic facilities must be satisfied by the objects of the application;
3. the non basic facilities must be satisfied by meta-objects;
4. the non functional capacities are added to an application object through its specific meta-objects, and;
5. the base-object can be structuraly and behavioraly changed at execution or compilation time.

In figure 1, we can see the reflection process in a computational system. The computational system is divided in two or more computational levels. The user sends a message to the computational system. The message is treated by the functional level, which is responsible for performing the work correctly, while the non functional level manages the work of the functional level.

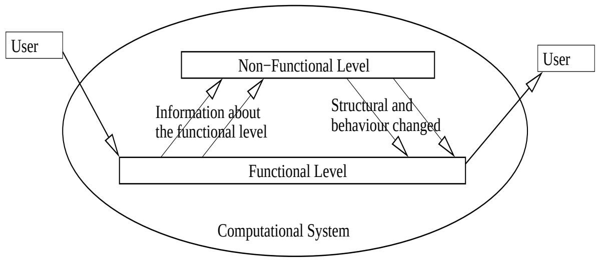

Fig. 1. Reflective Computing System Overview

With this ability a system is more capable of changing and adapting its structure and behavior. By separating the base-level functions from the meta-level functions we increase reusability, diminish complexity, and make the system more flexible overall.

Computational reflection can be implemented by functional, logic or imperative programming. However, it is with the natural flexibility of the object model that computational reflection has shown its effectiveness and elegance in the resolution of programming problems [13].

In computational reflection, classes, methods, attributes and objects representations are redefined by meta-classes and meta-objects. The level in which the meta-classes and meta-objects are available is called meta-level.

4 Proposed architecture

We propose an architecture for adaptive applications that use the meta-level and reflection concepts. The base-level contains all the basic functionalities (component 1 - figure 2) while the meta-level holds all the objects that control the creation and adjustment of the user models and stereotypes (component 2 - figure 2).

The meta-level has enough information to create and adapt the user models, stereotypes or any other structure necessary to adapt the system to the user. This is possible because the intersection concept (point (a) - figure 2) provides all the base-level system data (structure, behavior and user behavior) and observations of user bahavior. In this way, machine learning methods (mainly classification) process the data and offer support for resolution of decision-based problems.

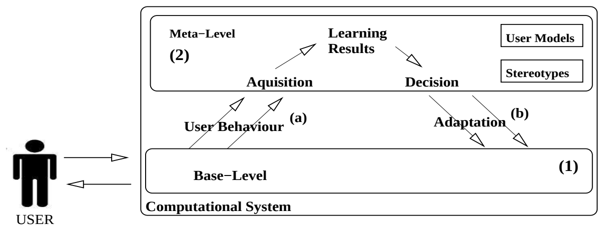

Fig. 2. Adaptive System Architecture

The decision can be used through the introspection ability (point (b) - figure 2) for changing the system functionalities (describes and executed by base-level) transforming the system and adapting it to the user's needs.

This architecture allows management of the functional aspects separately from the ones related to the acquisition and maintenance of the user's models. This is possible because the interface between base-level and meta-level is unique. Meta-level components can be removed and added to the architecture without modification to the base-level.

5 Application Example

To verify the adherence of the meta-levels architecture in adaptive systems, we implemented an adaptive system that uses the proposed architecture.

In this example a book store system (client and server) is implemented. The system is accessed by mobile equipments (cell phones) that use J2ME (Java Micro Edition) technology [14].

When the user is placing a book order using his cell phone, the following steps are executed:

1. he is identified by the cell phone number;
2. he searches the book by the book's name or by the author's name in the book's catalog;
3. he selects the desired book or books;
4. he adds the book to his “shopping cart”;
5. he executes another search and adds more books to his “shopping cart”;
6. he confirms the purchase, and;
7. he receives a message to confirm the purchase.

To attend the customers, the server side is implemented using the Java Servlets Technology [14]. Basically, the server is capable of keeping the “shopping carts” of the users, answering queries and finalize purchases. The book's database is written in XML. The system architecture can be visualized in figure 3.

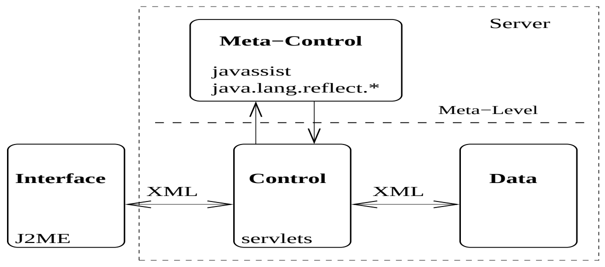

Fig. 3. Architecture of experiment

5.1 Meta-Level

While the base-level is responsible for all functionalities described before, the meta-level is responsible for the acquisition of the user's behavior beyond the computational system. Through the use of that information the meta-level changes the behavior of the base-level with the intention of adapting the system to the user.

Basically, the implemented features for this prototype are: interface changes that adapt the presentation or the navigation path, and; a list of recommended items for each customer (like typically done by Amazon [15]).

During the adaptation of the interface, the text components on authors and books are changed - removing, inserting or ordering the components according to the users' preferences. In the adaptation of the navigation, links are shown or hidden according to users' preferences. The recommendation algorithms use input about a customer's interests to generate a list of recommended items.

In both cases, the acquisition and maintenance of the user model is carried out through the observation of the user behavior. This is determined by means of the user's clicks and purchases.

The experiment showed that it is possible to reuse the components (i.e. the presentation adaptation component, the navigation adaptation component and the recommendation algorithms). The only action that must be taken is to change the components in the meta-level.

6 Conclusions

This paper described a meta-level architecture to separate the components responsible for acquisition and manipulation of the users' models from the other components of an adaptive system.

The architecture proposed here can be used to add the ability of adaptation to stable computational systems. It also allows management of the functional aspects separately from the ones related to the acquisition and maintenance of the user's models.

To validate this architecture, a system with adaptive interface was implemented. Through this experiment, it was verified that it is possible to reuse the components responsible for the user modeling only by adding them as stable meta-level components. Moreover, the need of tools to implement the computational reflection concept was identified. In the example, the meta-control component had to be built in the server side. This was because the technology used to implement the clients (J2ME) does not implement the computational reflection ability.

As future work, we plan to verify the applicability of this architecture in other kinds of adaptive systems, for example: adaptive information retrieval, recommendation systems and others.

Acknowledgments

The authors would like to thank Lavínia Boucault Napoleão and Luiz Carvalheira for all their remarks and cooperation in this paper.

References

[1] Palazzo, L.A.M. (2002) Sistemas de hipermídia adaptativa. In: JAI 2002 - XXI Jornada de Atualizacão em Informática, http://gpia.ucpel.tche.br/∼lpalazzo/sha/

[2] Webb, G.I., Pazzani, M.J., Billsus, D. (2001) Machine learning for user modeling. User Modeling and User-Adapted Interaction 11: 19–29

[3] Danilowicz, C., Nguyen, H.C. (2002) Using user profiles in intelligent information retrieval. In Hacid, M.S., Rs, Z.W., Zighed, D.A., Kodratoff, Y., eds.: Foundations of Intelligent Systems. 13th International Symposium. Number LNAI 2366, Lyon, France, Springer-Verlag 223–231

[4] Rich, E. (1999) Users are individuals: Individualizing user models. International Journal of Man-Machine Studies 51: 323–338

[5] Papatheodorou, C. (2001) Machine learning in user modeling. In Paliouras, G., Karkaletsis, V., Spyropoulos, C.D., eds.: Machine Learning and Applications. Number LNAI 2049. Springer-Verlag Berlin Heidelberg, Berlin 286–294

[6] Orwant, J. (1995) Heterogeneous learning in the doppelganger user modeling system. User Modeling and User-Adapted Interaction 4: 107–130

[7] Fink, J., Kobsa, A. (2002) User modeling for personalized city tours. Artificial Intelligence Review 18: 33–74

[8] Kobsa, A. (2001) Generic user modeling systems. User Modeling and User-Adapted Interaction 11: 49–63

[9] Pohl, W., Nick, A. (1999) Machine learning and knowledge representation in the labour approach to user modeling. Proceedings of the Seventh International Conference on User Modeling. 179–188

[10] FERBER, J. (1989) Computational reflection in class based object-oriented languages. SIGPLAN Notices. 24: 317–326

[11] LISBÔA, M. (1997) Arquiteturas de meta-nível. Tutorial XI Simpósio Brasileiro de Engenharia de Software 1: 210-298

[12] WU, S. (1997) Reflective Java: making Java even more reflexible. http://www.ansa.co.uk, Cambridge, UK.

[13] KICZALES, J.G.R., BODROW, D. (1991) The art of the metaobjects protocol. MIT Press, Cambridge

[14] SUN (2004) Java Technology. http://java.sun.com

[15] Linden, G., Smith, B., York, J.: (2003) Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Distributed Systems OnLine 1: 24-26