Meta-Level Constructs in Content Personalization of a Web Application

Annappa B, K Chandrasekaran, K C Shet
Department of Computer Engineering, National Institute of Technology, Karnataka, India
annappa@ieee.org, kchnitk@ieee.org, kcshet@rediffmail.com

***Abstract* - In today's business environment, web applications become more and more complex but they still need to be flexible for changes, easy to maintain and the development life cycle need to be short. A reflective technique seems to be the best way to achieve flexibility of the web applications when adding the personalization features like recommendations, special offers, etc. Most of the algorithms help to achieve personalization; but, little attention has been paid to the design and modeling process of internet applications. Personalization will help to cope with increasing complexity of Business Enterprise level Applications. High-level, cleanly layered solutions open up promising possibilities to overcome these difficulties. This paper gives an insight into the content personalization of a web application using meta-level constructs.**

I. INTRODUCTION

Personalization has become a very important issue in software applications like web applications. The personalization has become important topic because of the popularity of the World Wide Web and evolution of hardware appliances. While designing personalized software, different concerns like modeling the user, implementing personalized interfaces, modeling customization rules etc. must be addressed and integrated. Extensive study has been carried out on user modeling, profile derivation and personalization algorithms but little attention has been paid to the designing and modeling of the personalized software [1]. Therefore, we feel that personalization is an important aspect in application's evolution. This paper describes the content personalization of web applications using the meta-level constructs. Here the term construct refers to the proxy interception pattern which will be placed at the meta-level to achieve the content personalization.

The meta-level architectures are based on the idea of computational reflection. Computational reflection is defined as the ability to observe and manipulate the computational behavior of a system through a process called reification. Reification is a technique which enables the system to maintain information about itself and use this information to change its behavior.

Meta-level architectures provide clear separation of system functionality and system behavior. The system functionality is represented by base level objects and behavior of the system is represented by meta-level objects. Meta-object protocol (MOP) establishes an interface among the base level and meta-level components. A MOP provides a high-level interface to the programming language implementation in order to reveal the program information normally hidden by the compiler or run-time environment. As a result, programmers can develop language extensions, and adapt component behavior or even make changes to the systems more transparently [2]. So, we approach the problem of content personalization in a web application using meta-level constructs.

The content personalization is classified into two types:

1) Node structure customization: Structure personalization usually appears in those sites that filter the information that is relevant for the user, showing only sections and details in which the user may be interested in. The user may explicitly indicate his preferences.

2) Node content customization: occurs when different users perceive different values for the same node attribute; this kind of content personalization is finer grained than structure personalization.

To achieve the personalization, Object Oriented Hypermedia Design Model (OOHDM) approach [3] [4] has been used. The key concept in OOHDM is that Web Application models involve a Conceptual, a Navigational Model and an Interface Model. The concern of the conceptual model is to represent domain objects, relationships and the intended applications' functionality. In the OOHDM approach, users do not navigate through conceptual objects, but through navigation objects (nodes) that are defined as views on conceptual objects. As Web Applications are considered as hypermedia applications, links connecting nodes are defined as views on conceptual relationships. Finally, the abstract interface model specifies the look and feel of navigation objects together with the interaction metaphor.

The structure of the paper is as follows: The section 2 describes the design for the content personalization of application using meta-level constructs. The section 3 illustrates the situation where the design can be applied. The section 4 describes the implementation of personalizers through proxies. The section 5 gives the implementation details of personalizers using Strategy Design pattern. The section 6 shows results obtained.

II. DESIGN USING META-LEVEL CONSTRUCTS

The design is based on the criteria of inserting interception layer between the objects which model the personalization logic in the personalization layer and the application logic in the application layer [5]. The fig. 1 shows the insertion of interception layer between personalization layer and the application layer.

978-1-4244-9034-/10/$26.00©2010 IEEE

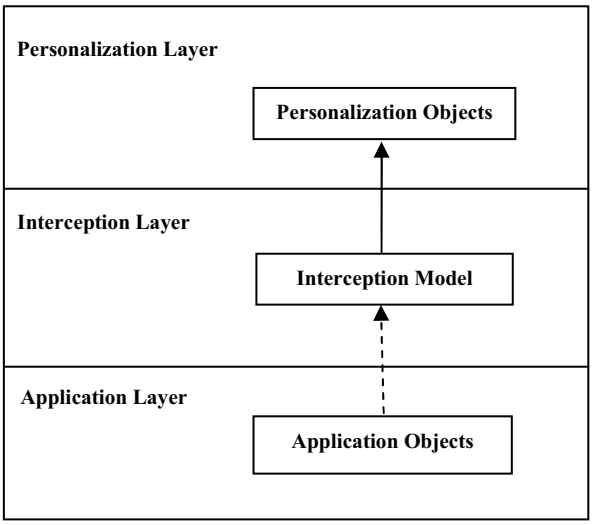

Fig 1: Layered architecture for Personalization

Here personalization objects and application objects are decoupled so that they can evolve in an independent fashion. Proxy interception pattern and Strategy design pattern are used in realizing the personalization [6]. In proxy interception pattern the idea is to put an object between the client and the real object, so that the client thinks it is sending messages to the real object, when the proxy is actually intercepting them and solving its execution by collaborating with the real object, and eventually other objects. In Strategy Design pattern the object and behavior are separated and put into different classes. This enables any algorithm to be used at any time. The interception layer is the meta-level construct to realize the personalization of a web application. The UML design of the proxy pattern and Strategy Design pattern are as shown in Fig. 2.a and Fig. 2.b.

III. IMPLEMENTATION OF PERSONALIZATION

We have implemented an electronic store that sells *products* to *customers* as a case study. The customer may put products in his/her *shopping cart*, indicating the number of items of each product. When he/she decides to buy, the *check-out* process generates an *order* containing the products, *paying mechanism, shipping address* and *delivering options* (whether the product should be gift-wrapped or not). Each customer has an *account* containing his buying history. The fig.3 presents an object model representing key features as classes.

Variations in the domain model such as adding new products or paying mechanisms can be solved by sub-classing and composing objects. Suppose, if we want to introduce some personalization capabilities, then there will be a problem. For example, we want to personalize product prices; one possibility is to let products delegate the price computation to the corresponding customer object as shown in Fig. 3.a. The same strategy can be use with other object attributes.

The problem with this approach is that it works well if no new customization rule is added or if we don't need different

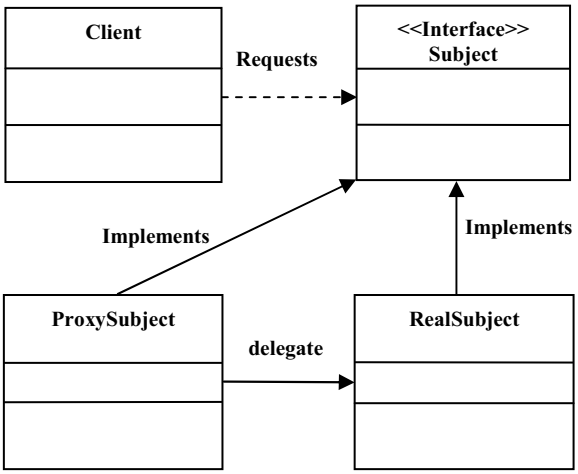

Fig. 2.a: UML design of Proxy Interception Pattern

rules in different usage contexts, e.g.: when the user browses the product, he gets one price (his price) but when he checks-out, the price may be personalized differently (for example taken into account a special offer when buying some combination of products). If this happens we will end re-designing core business classes constantly and sooner or later the code will become messy.

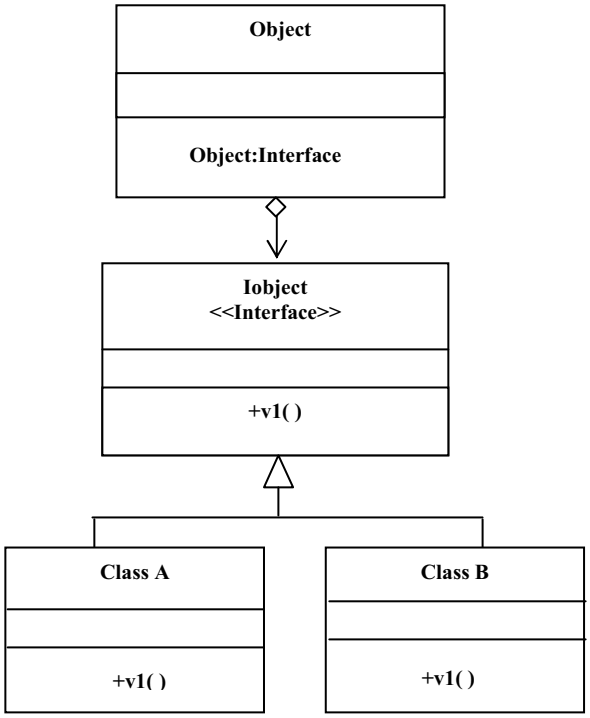

Fig. 2.b. UML design of a Strategy Design pattern

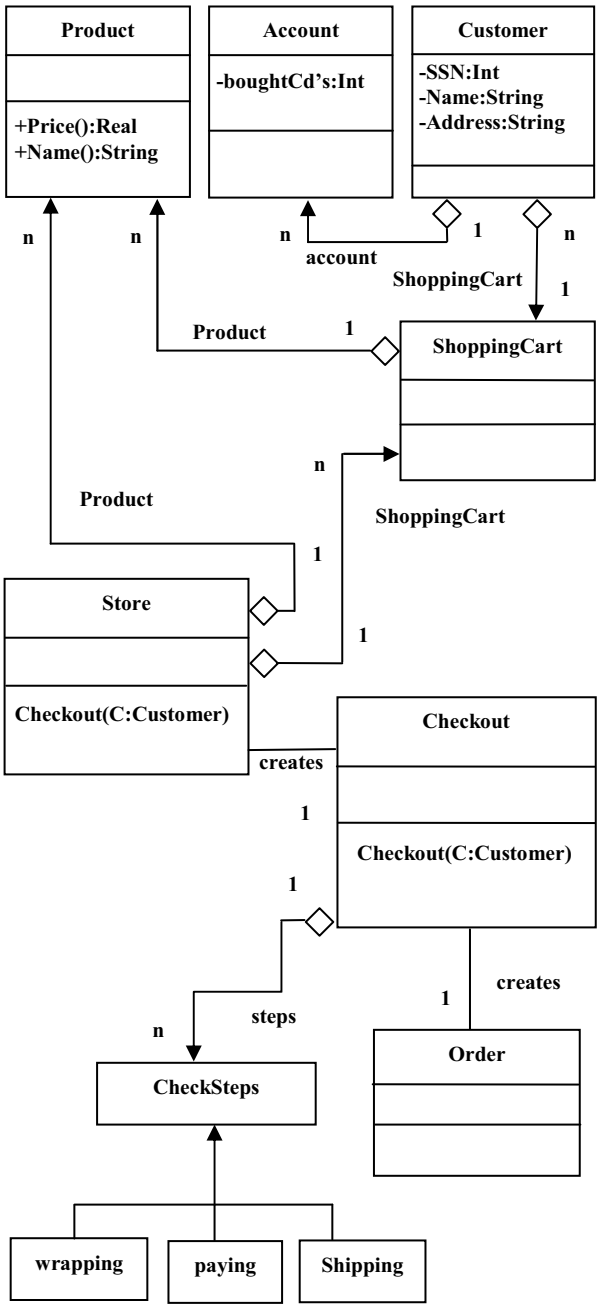

Fig. 3: Conceptual model of the e-store

IV. IMPLEMENTATION OF PERSONALIZERS THROUGH PROXIES

The solution to the personalized price problem is based on the proxy interception pattern [6], which is used to make changes to a given object without altering the core functionality. For each class to be personalized a corresponding proxy object is defined and it is connected to the real object in an instance basis. The solution is shown in Fig. 4.a.

Fig. 4.a describes the concept of proxies. It consists in

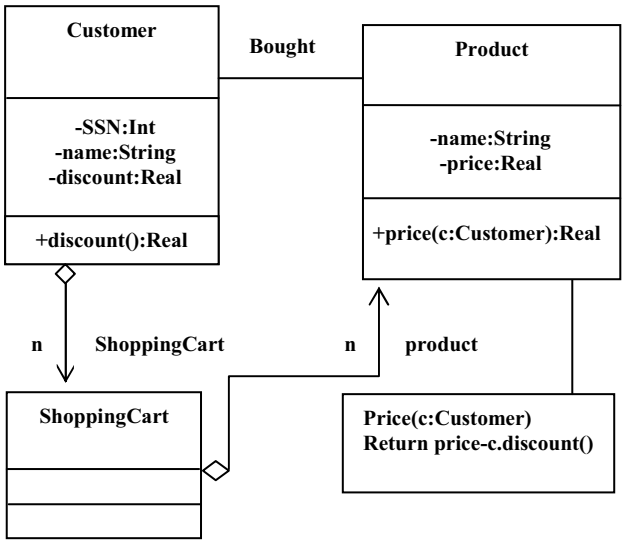

Fig. 3.a.Hard-Coding personalized prices in products

intercepting messages and processing them without intervention of the real object and returning the result. Following with the example of the product's price, the proxy traps the message *userid,* and then communicates with the real object to get the absolute discount for that particular user. Then, after getting the absolute discount the proxy will calculate the personalized discount. Fig. 4.c gives the idea of discount algorithm implementation. The invoke () method will get the absolute discount for a user and then the getdiscount() method will calculate the personalized discount corresponding to the user and finally summation of the absolute discount and the personalized discount will returned by the proxy.

V. IMPLEMENTATION OF PERSONALIZERS THROUGH STRATEGY DESIGN PATTERN

The implementation of recommendations, which is a popular kind of personalization feature, is done using a Strategy design pattern [6]. Different recommendation

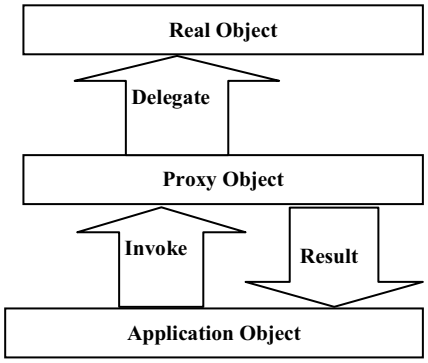

Fig. 4. a. Functionality of a proxy object

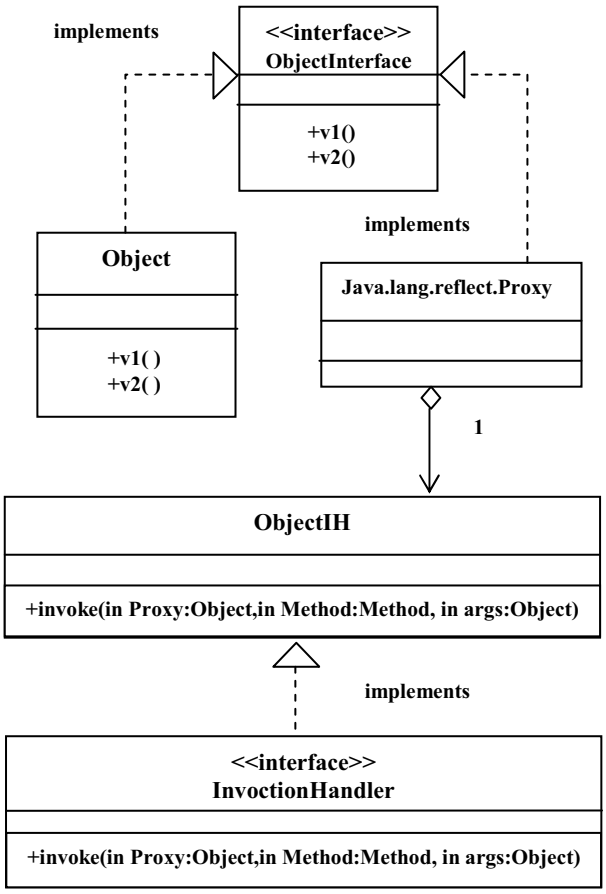

Fig. 4.b. UML design of the proxy pattern

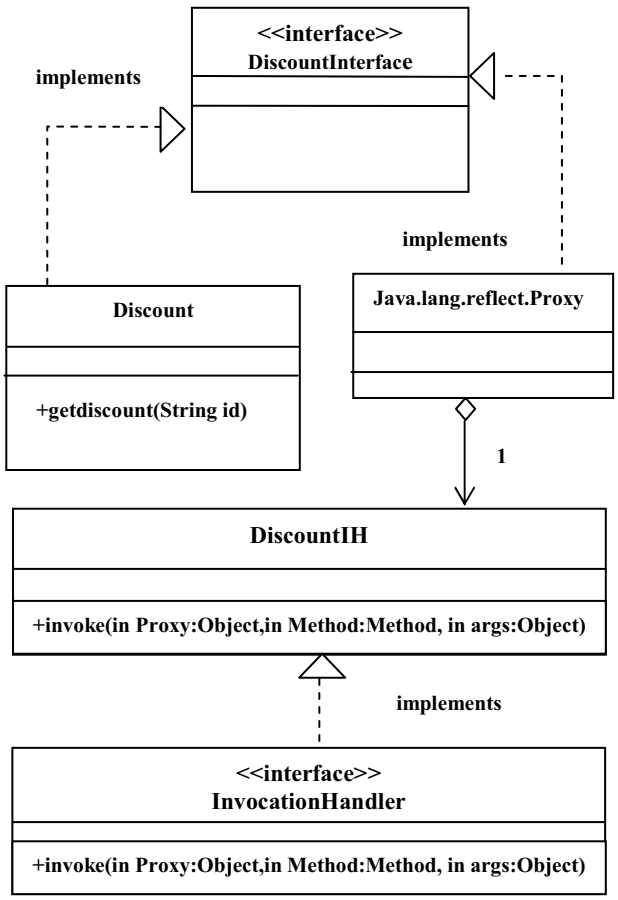

Fig. 4.c. Discount algorithm implementation using proxy

algorithms are organized in a class hierarchy and thus we can switch algorithms dynamically. This is not possible if we implement the recommendation as a method in class Store. This is shown in Fig. 5. Recommendation algorithms will thus interact with customer objects and their accounts to obtain the desired results.

VI. RESULTS

There are various design patterns [6] to achieve the web content personalization. But in our implementation of electronic bookstore, generic proxy interception (at meta-level) has been used. The proxy pattern captures the method invocation without cluttering the application code and does the job of the real object. For example, Fig. 6 gives the clear idea of the above discussed example of price customization. Similarly, a personalized recommendation that has been achieved is as shown in the Fig. 6.a. and Fig. 6.b. The main advantage of placing the meta-level concept is that the user can add or remove the personalized behaviors to existing applications in a non-intrusive way. Decoupling the interface and the implementation of classes and objects helps to make the content personalization manageable and scalable.

VII. CONCLUSION

In this paper, we have presented a design approach for building personalized e-commerce applications. This approach is based on clear separation of concerns like core business logic, user profile and personalization rules. The design components related with personalization are not hard coded in core application objects, but built as separated objects using meta-level constructs (interception patterns). By applying well known patterns (such as Proxy, Strategy) the e-commerce applications can be seamlessly extended with the personalization features. The approach is non-intrusive because it provides mechanisms for adding extra functionality (e.g. personalization) without changing the core business logic.

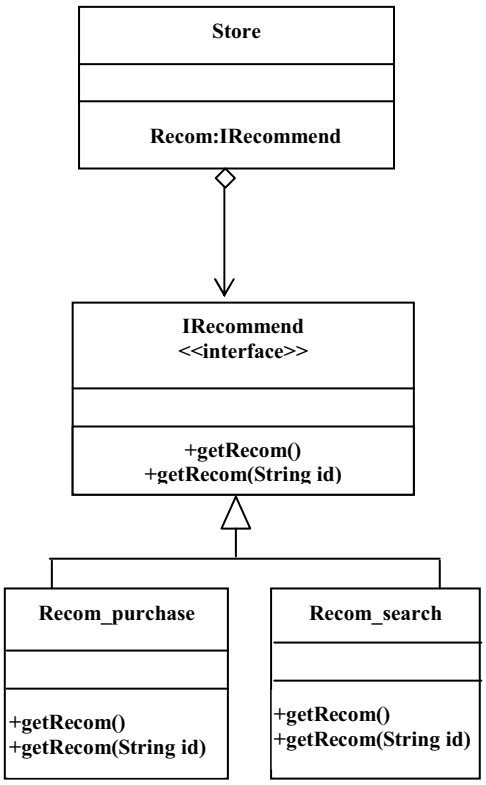

Fig. 5. Recommendations implementation using Strategy Design pattern

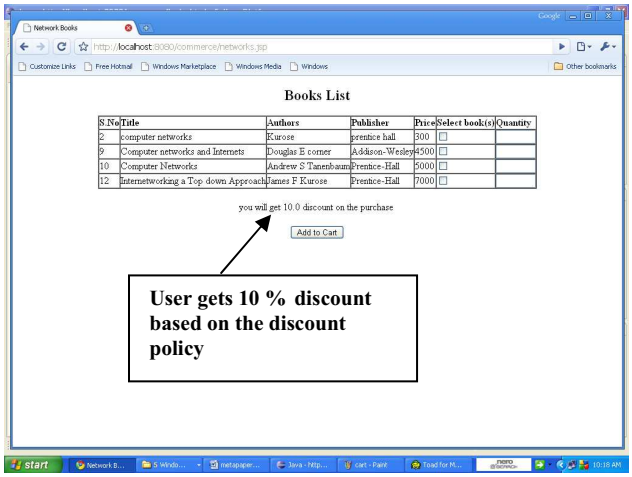

Fig. 6. Price customization according to the user.

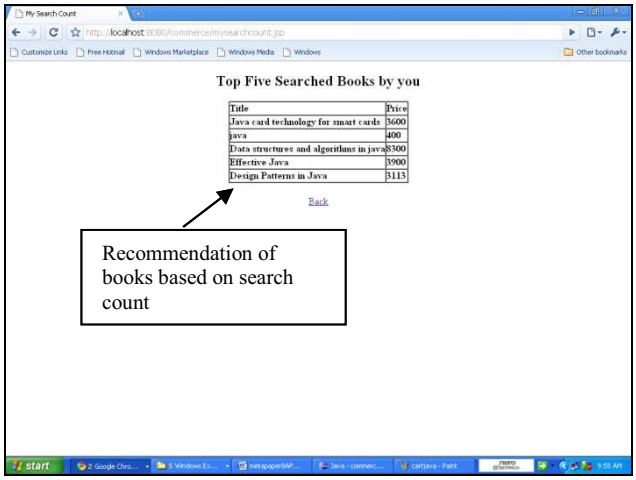

Fig. 6.a. Recommendations based on the search count of the user.

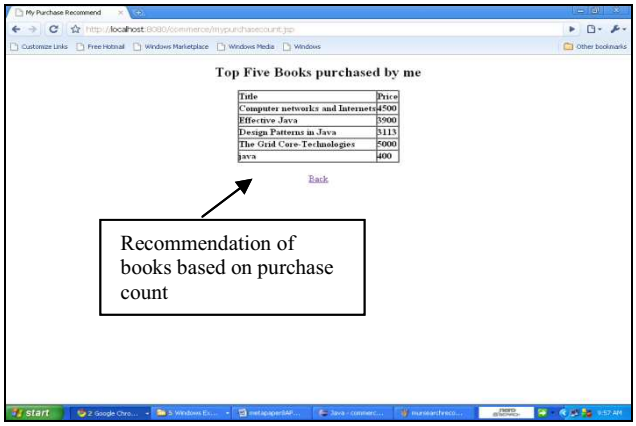

Fig. 6.b. Recommendations based on the purchase count of the user.

REFERENCES

[1] Andrés Fortier, Gustavo Rossi, Juan Cappi, "Using Meta-Level Techniques to Personalize O-O Applications", Workshop on Engineering Complex Object-Oriented Systems for Evolution, OOPSLA 2001.

[2] Alessandro F. Garcia, Delano M. Beder and Cecilia M.F. RUBIRA, "A Unified Meta-Level Software Architecture for Sequential and Concurrent Exception Handling", *Computer Journal 2001* Volume 44, Number6, pp.569-587.

[3] Juan Cappi, Gustavo Rossi, Andrés Fortier, Daniel Schwabe "Seamless Personalization of E-Commerce Applications", 2nd International Workshop on Conceptual Modeling Approaches for e-Business, *Springer Verlag*, December 2001.

[4] Object Oriented Hypermedia Design Model http://www.telemidia.puc-.br/oohdm/oohdm.html/Design workshop at HT'08

[5] Andrés Fortier, Juan Cappi, Gustavo Rossi, "Interception Patterns", Proceedings of PloP 2003.

[6] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns. Elements of Reusable Object-Oriented software*, Addison Wesley 1995.

[7] Using Java Reflection, http://java.sun.com/developer/technicalArticles/ALT/reflection/index.html

[8] Gustavo Rossi, Daniel Schwabe, Robson Guimarães, "Designing Personalized Web Applications", *Proceedings of the 10th International Conference of the WWW*, May 2001

[9] Adrian Lienhard, Mewa:"Meta-level Architecture for Generic Web-Application Construction", Software Composition Group University of Bern, Switzerland November 17, 2003.