

# Towards the Identification of Concerns in Personalization Mechanisms Via Scenarios

Cláudia Mesquita, Simone Diniz Junqueira Barbosa, Carlos José Pereira de Lucena  
Informatics Department, PUC-Rio  
R. Marquês de São Vicente, 225  
Gávea – Rio de Janeiro – RJ 22453-900  
Brazil  
e-mail: {mesquita, sim, lucena}@inf.puc-rio.br  
+55 21 3114-1500 ext. 4430

## Abstract

In this position paper we discuss the identification of concerns related to personalization mechanisms. We propose using scenarios to extract systems requirements, user's task, and related features. We argue that if we clearly identify the personalization-related concerns, we could add a posteriori personalization features to legacy systems, following an advance separation of concerns such as aspect, subject and so on.

## Introduction

There have been some attempts to identify aspects at the early stages of the software life cycle [Tekinerdogan et al., 1998; Grundy, 2000]. These approaches aim to achieve some of the desired software engineering quality factors, such as reducing adaptation, evolution and maintenance costs. In this position paper, we are interested in identifying early aspects in personalization systems.

It is a known fact that the Internet has brought about deep changes in the way people work and in the way software designers conceive and develop computational artifacts. One interesting evolution is related to the adaptation mechanisms involved in personalization. In a nutshell, “personalization is about building customer loyalty by building a meaningful one-to-one relationship; by understanding the needs of each individual and helping satisfy a goal that efficiently and knowledgeably addresses each individual's need in a given context (...) it is about the mapping and satisfying of a user's/customer's goal in a specific context with a service's/business's goal in its respective context” [Riecken, 2000]. Although e-commerce has been the driving force underlying the increase in personalization features, many applications profit from the ongoing research and technological advances in this area. We adopt here Blom's definition of personalization, as “a process that changes the functionality, interface, information content, or distinctiveness of a system to increase its personal relevance to an individual.” [Blom, 2000].

The purpose of this position paper is to suggest that advanced separation of concerns (AsoC) approaches may help specify personalization solution and point further investigation about whether personalization-related concerns may be identified early in

the development process and, if so, how it may impact software design, flexibility and maintainability. In this position paper, we will utilize scenarios of use that depict some personalization characteristics as a resource to try to extract concerns as early aspects.

If we are able to clearly identify personalization-related concerns, we will be able to assess the feasibility of adding *a posteriori* personalization features to existing applications, following an advanced separation of concerns approach such as aspect-oriented programming [Kiczales et al, 1997], subject-oriented programming [Harrison et al, 1994], composition filters [Aksit et al, 1994] and so on.

## **Human-Computer Interaction and Personalization**

Many concerns underlying personalization are directly related to the research area of human-computer interaction (HCI). Design of personalization artifacts should be driven by users' goals and needs, not by the novelty of cool tools [Kramer et al., 2000]. Karat et al. put it nicely: "Every tool should feel like it was custom designed for you, the user in your context." [Karat et al., 2000]. Usability evaluation methods and tools have traditionally assessed the quality of use of an interactive system. Usability issues may be thought of as cross-cutting concerns which drive design. A typical set of usability concerns is as follows [Nielsen, 1993]:

- ease of learning and understanding: How long does it take for users to learn how to use an application? After a period of not using an application, can users still remember how it works?
- ease of use: Does the application support the user's tasks adequately? Are there many possibilities of making mistakes? Does the application provide adequate and timely feedback?
- user control: Can the users predict what will happen when they trigger an action? Can they interrupt lengthy processes?
- flexibility: Are there alternative ways to perform a task and achieve a goal?
- efficiency: How long does it take to perform a task? How long does it take to recover from an error? Does the application succeed in improving the user's productivity?

These concerns are interrelated, and design decisions usually involve cost/benefit analysis and tradeoffs. For instance, applications for highly specialized users may provide ease of use at the expense of more extensive training needs. Flexible applications may provide both ease of use and ease of learning, possibly by means of two or more distinct interaction paths leading to the same result, i.e., achieving the same goal.

How usability concerns affect the resulting design and implementation and propagate throughout design is the focus of HCI research. We will address in this paper the identification of concerns as related specifically to personalization mechanisms, and leave the relationship between these concerns and HCI concerns to future research. Filman has stated that AOP is useful for handling systematic requirements/concerns<sup>1</sup>

---

<sup>1</sup> " Systematic concerns/requiments pervade the behavior of the system, but can be realized by 'doing the right thing' in 'all the right places'" [Filman, 1998].

[Filman, 2001]. An interesting issue would be whether (and if so, which) usability concerns might be considered systematic concerns.

### **Domain-Separable Analysis of Personalization**

In our first attempt to identify personalization concerns, we have found three levels of concerns that may be analyzed independently of problem domain, i.e., as a solution domain.

- **syntactic level** (domain-independent): those mechanisms that do not involve domain information, but only users' (as a whole) activity in the application. For instance, the list of the  $n$  most recent items viewed or inserted, the list of  $n$  most visited items, and so on;
- **semantic level** (domain- and user-dependent, but task-independent): those mechanisms that make use of relations between entities in the domain, such as those represented in a domain ontology, taking into account user information. For example, the list of items related to the user's (as an individual) interests;
- **pragmatic level** (domain-, user- and task-dependent): those mechanisms that take into account information which not only has a correspondence to the conceptual domain, but also to the users' tasks. For instance, the user's agenda of tasks that should be carried out at the moment, a list of tasks that are late, or a list of users whose tasks are affecting the current user's work.

Later stages in the design will need to take into account domain-specific information, but in our research we will try to keep our analysis domain-independent, considering two complementary views of an application: a content view and a process view.

In the **content view**, users' goals are not task-related, but content-related. In other words, users need only to find some information, by whichever means available in the application. This view represents the portions of an application designed for information-retrieval, for instance. In the **process view**, the focus are the tasks users need to perform using the application. The content is important only inasmuch as it affects the task at hand. This view is the predominant view of workflow-related systems.

The content view may present personalization at both syntactic and semantic levels, but not at the pragmatic level. This view has no representation of users' tasks. The process view, on the other hand, is more comprehensive, in that it includes both task and content representations. Due to lack of space we won't be able to address both of them here. We will present only a simple scenario related to the content view.

### **Sample Scenarios**

To help identify concerns early in the design process, we propose using scenarios [Carroll, 1995], from which we may extract systems requirements and users tasks and related features. We present a scenario that illustrates possible usage situations of applications without personalization, and suggest some directions for adding personalization features to them, while discussing the corresponding concerns.

### ***Scenario I: Information retrieval application***

John needs to find a book to study for his final exams on Non-Euclidean Geometry next week. He enters the university library website. Since the book is dated 10 years ago, he doesn't check the list of recent acquisitions. He proceeds to browse the index of Mathematics and is led to another index. He notices it may take a while to find the book that way and, since he knows the name of the book he is looking for, he decides to search the book by providing the author's last name. The system presents summary information of 12 books, and he quickly locates the book he wants. He examines the book details, in order to see in which section in the library the book is located. He exits the system and rushes to the library to get the book.

In this typical information retrieval system, we identify, by means of the scenario, the following features: *browsing through a structure of indexes*, *searching a database*, *looking into a list of highlighted items* (in this case, recent acquisitions). All of the aforementioned usability concerns are at play here: ease of learning (the user shouldn't need training); ease of use (the user shouldn't make mistakes because of bad interface design); user control (the user may change his/her mind during interaction and interrupt the current task); flexibility (there are alternative ways to reach a goal); and efficiency (the user finds information faster by using the system than without it).

Without personalization, every user is presented with the same content and structure. The following scenario illustrates a simple feature addition to the system in order to provide a small degree of personalization:

John has decided to do some research on Software Engineering, and needs to find a few books that may be important for his work. The first book he finds is a classic, and he must read it. Fortunately, the new version of the library system presents a mechanism for *bookmarking* the books he finds interesting, and John decides to try this new feature right away. He looks up 8 or 9 books about Software Engineering, 3 of which he decides to bookmark. He checks the items that have been bookmarked, and decides to start from "the classic". He examines the information of that book's location in the library, exits the system and goes to the library to check that book out.

With such a customization mechanism, the library homepage can supply a section called something like "my favorite books", which presents a list of books that have been bookmarked by the current user. One should notice that, by adding this bookmarking feature, another user goal emerges: the user will need now to be able to manage his/her bookmarks.

Upon inspecting the scenario, we have identified the following requirements:

- user control of the adaptation: in customization mechanisms such as the one presented here, the user has total control of the adaptation. In terms of Kühme's classification [Kühme et al. 1993], the user is responsible for initiating the adaptation process,

deciding whether or not to adapt, and executing the adaptation (adding the current item to the bookmark list)

- persistence: the information of this session should be stored for later recall in future sessions with the application
- user model (user's bookmarks): the personalized information is related only to the individual user responsible for the customization

Another kind of personalization is via recommendations, where a list of items considered important is presented to the user at certain points during the interaction. This list may be inferred by the system based on the user's profile, his/her behavior during interaction with the system, and even on other (somehow correlated) users' profiles. In our library example, the system could infer a set of relevant books based on the user's interaction history over the last few sessions.

The following requirements are identified in this part of the scenario:

- system control of the adaptation: the system takes initiative, decides and executes the adaptation, which corresponds to the creation and maintenance of the list of recommended items
- monitoring: the system monitors the users' activities within the system
- user model (interaction history): the individual user's activities within the system provide input to the personalization mechanism
- persistence: the information of this session should be stored for later recall in future sessions with the application

Personalization mechanisms are typically scattered in software design. These mechanisms involve some concerns already identified as suitable for AOSD: log history, monitoring, tracking, and persistence, to name a few. We believe personalization may be analyzed and specified using ASoC approaches [TRESE, 2002].

## **Conclusions and ongoing work**

As software evolves, new concerns may arise. In this position paper, we have taken a first step towards an analysis about whether early aspects will improve the ability to "remodularize" software according to new concerns in a non-invasive way and without eliminating existing encapsulation, i.e., encapsulation derived from previously identified concerns.

An issue that deserves further investigation is whether the concerns we have identified as crosscutting concerns at the requirements stage would be propagated as an aspect throughout design, i.e., if the aspectual composition would be maintained at the design and implementation levels.

As for the early stages of requirements analysis and design, we need robust methods to help identify crosscutting concerns from scenarios, as well as the relationships between these concerns.

With respect to personalization systems, we need to develop a real case study to investigate how the identification of early aspects might affect both the design process and the resulting design. We will create scenarios of use that depict more sophisticated personalization mechanisms, and scrutinize them carefully, in order to derive representations of users' tasks and system requirements, together with the corresponding concerns, if any. Having captured this information, we will investigate how the identified concerns will propagate throughout the design process.

## Acknowledgments

Our thanks to Christina von Flach and Alessandro Garcia for comments to drafts this position paper.

## References

- Aksit et al., 1994 M. Aksit and K. Wakita and J. Bosch and L. Bergmans, Abstracting object interactions using composition filters, Lecture Notes in Computer Science, 1994.
- Blom, 2000 Blom, J. "Personalization – a taxonomy". In *Proceedings of the CHI 2000 Workshop on Designing Interactive Systems for 1-to-1 E-commerce*. Online at <http://www.zurich.ibm.com/~mrs/chi2000/> (last accessed Mar/2002).
- Carroll, 1995 Carroll, J. (ed) *Scenario-based Design: Envisioning Work and Technology in System Development*. New York, NY. John Wiley and Sons, 1995.
- Grundy, 2000 Grundy, J.C. Multi-perspective specification, design and implementation of components using aspects, International Journal of Software Engineering and Knowledge Engineering, Vol. 10, No. 6, December 2000, World Scientific
- Harrison et al, 1994 Harold Ossher, William Harrison, Frank Budinsky, and Ian Simmonds, *Subject-Oriented Programming: Supporting Decentralized Development of Objects*, Proceedings of the 7th IBM Conference on Object-Oriented Technology, July, 1994
- Karat et al., 2000 Karat, J.; Karat, C-M.; Ukelson, J. "Affordances, Motivation, and the Design of User Interfaces". In *Communications of the ACM* 43, 8. August, 2000. pp.49–51.
- Kiczales et al., 1997 G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin, Aspect-Oriented Programming. In proceedings of ECOOP '97, Springer-Verlag LNCS 1241. June 1997
- Kramer et al., 2000 Kramer, J.; Noronha, S.; Vergo, J. "A User-Centered Design Approach to Personalization". In *Communications of the ACM*, 8. August, 2000. pp.45–48.
- Kühme et al. 1993 T. Kühme and M. Schneider-Hufschmidt. Introduction. In M. Schneider-Hufschmidt, T. Kühme and U. Malinowski. *Adaptive*

- User Interfaces: Principles and Practice*. North-Holland, 1993, pp.1-9.
- Nielsen, 1993 Nielsen, J. *Usability Engineering*. Academic Press, 1993.
- Riecken, 2000 Riecken, D. "Personalized Views of Personalization". In *Communications of the ACM* **43**, 8. August, 2000. pp.27–28.
- Tekinerdogan et al., 1998 B. Tekinerdogan and M. Aksit, Deriving *Design Aspects from Canonical Models*, in *Object-Oriented Technology*, S. Demeyer and J. Bosch (Eds.), LNCS 1543, ECOOP'98 Workshop Reader, Springer Verlag, pp. 410-413, July 1998.
- Trese, 2002 TRESE Aspects and advanced separation of concerns homepage. [http://trese.cs.utwente.nl/aspects\\_asoc/index.htm](http://trese.cs.utwente.nl/aspects_asoc/index.htm) (last accessed Feb/2002).