

A Modelling Method for Designing Adaptive Hypermedia

Toni Alatalo
Toni.Alatalo@oulu.fi

Janne Peräaho
Janne.Peraaho@oulu.fi

Department of Information Processing Sciences, University of Oulu
OWLA research group, <http://owla.oulu.fi/>

Abstract

This position paper presents a novel method for modelling adaptivity in hypermedia design specifications. The method is based on an approach for designing adaptive hypermedia that integrates adaptivity in the structure of the system, recognizing some objects as adaptors and developing heuristics based on what the content and functionality may adapt. The heuristics are captured in so-called shadows that may hide the potentially complex underlying algorithms from the designer. The method is intended to complement existing methodologies, such as OOHDM, which currently lack means for visualizing adaptivity aspects. An user-modelling centric example is presented, accompanied by an implementation on-line for the interested to try. Current limitations and directions for future work on the method are recognized.

Introduction

Several methods for modelling hypermedia as objects have been proposed, such as OOHDM (Schwabe&Rossi 1998), OO-HMethod (Gómez et al 2000) and an UML extension (Baumeister et al 1999). Of the existing methods, only the adaptive hypermedia design method (AHDM) supports adaptive hypermedia by including the design of user models within the methodology (Koch 1998). But even the AHDM approach does not support *the modelling of adaptivity dependencies when specifying the content and functionality of the hypermedia application*. That is, the previous modelling methods do not have means for the designer to visualize e.g. how certain features of the system depend on certain properties of the user. Here we present a modelling method for adaptive hypermedia, proposing a solution to that problem. The proposal is preliminary, and has not been thoroughly analyzed or put to practice yet.

De Bra, Eklund et al (1999) define adaptive hypermedia as a collection of nodes and links, that is accompanied by user profiles, which are used to adapt the presentation. Also Brusilovsky (1996), in his authoritative definition, specifies having a user model as the criteria for a hypermedia system to be adaptive. In our research, we attempt to approach the issue more broadly, in a more general way. While adapting to user is central in many applications, in our view it is not all there is to adaptivity, and in some cases not even a requirement for adaptive hypermedia. For example, considering mobile technologies, there are several different kinds of devices with very different user interfaces and adapting the hypermedia based

on device profiles - but not necessarily regarding any user profiles at all - makes sense. On the other hand, for example security aspects - while clearly having to do with user profiles - are not necessarily tied to the (psychological?) human characteristics often emphasized in adaptive hypermedia, but e.g. classifications based on organizational structures.

Thus, here adaptive hypermedia is understood in a more general sense: as hypermedia, that is adaptive respect to something(s), which depend(s) on the application domain. This is the basis for our view on designing adaptivity. In this paper, however, the modelling method is presented with an user-modelling centric example, demonstrating how the notation can be used to illustrate dependencies between user specific properties and behavior of the system. In this respect our approach differs from e.g. the AHAM Adaptive Hypermedia Architecture Model (De Bra, Houben & Wu 1999) in that we integrate the adaptivity in the content/functionality specification.

Also, we are aware that there is another kind of adaptivity (in the broad sense of the word) that our approach does not currently address. In our preliminary sketches we have separated between *adapting to difference* (e.g. to different users, which is the focus here) and *adapting to change* (e.g. changes in the whole systems as time goes) but in this position paper, that discussion is left out to be dealt with later. An example of a method for dealing with adapting to change can be found in the Demeter method for adaptive programming (Lieberherr 1996).

The rest of the paper is structured as follows: first, an overview of the design approach is given in the following section, then the actual modelling method is illustrated with examples, and the research position and future direction summarized in the final section.

Designing adaptive hypermedia

The modelling method under development is based on an adaptation concept which we call structural adaptation. Structural adaptive system consist of three parts (or group of system components), namely Adaptors, Heuristics, and Transformants. They can be understood as specific roles some objects in the system have with respect to adaptivity.

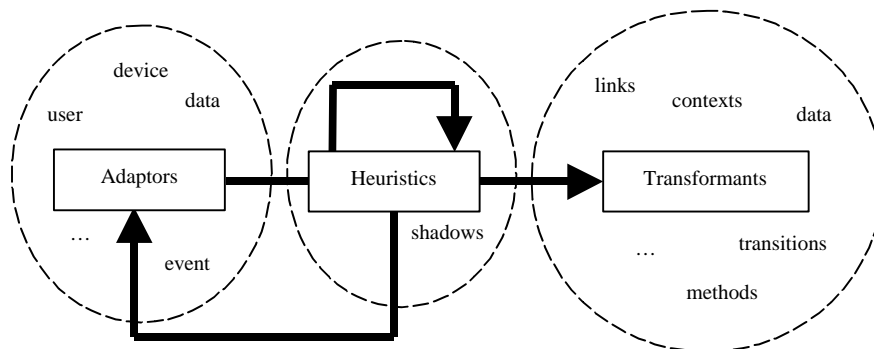


Figure 1. Adaptive system

Adaptors are system components to which the rest of the system adapts. Adaptors are defined by the application domain, system requirements and/or by software designers, so they can be anything: users, (client) devices or events, to mention just a few of them.

In order to be adaptive system's adaptors have to possess some information or attributes by which the system's adaptive components have a child-like relation. (Note that along with attributes adaptor can contain methods, too.) We call the information attached to the adaptor as adaptor's *properties*.

Transformants are system components which adapts to adaptors according to heuristics. Transformants consists of non-adaptive parts (skeleton) and adaptive parts (transformants), which are connected to non-adaptive parts through heuristic rules.

Because we are dealing with hypermedia bounded design method, one can identify and name transformants accurately. Transformants can be based upon data (related to application domain), links, contexts, transitions (links between contexts), methods, adaptors, and heuristic rules. Adaptors and heuristic rules acting as transformants is quite rare, but still possible and justified in special occasions.

Heuristics defines two-fold relations between adaptors and transformants: 1) in which adaptor's options transformant has relation to (variables and constants) and 2) what is the nature of the relation (mathematical and logical dependency).

Heuristics can be classified. Classification is based on adaptors which the heuristic rule is mainly involved. We call these heuristic classes as *shadows*. Thus one can speak of and name several shadows: user shadows, device shadows, and many more, depending on number of adaptors and shadows' objectives dictated by application domain, system requirements and/or system designer.

The shadow consists of one clause (heuristic rule), containing variables, constants, mathematical- and logical operands, which defines the relation of one or several adaptors and transformants. As a result of this each shadow has a binary output value, 0 (false) or 1 (true), which indicates adaptor's adaptive part's inclusion or disjunction. In other words, shadows acts as a logical glue which holds the transformant's skeleton and transformants together. Without these links adaptive system can not exists.

The design process of an adaptive system can be divided in six tasks based on concepts of hypermedia and adaptivity:

- design of adaptors
- design of shadows
- design of the information content
- design of contexts
- design of a navigation
- design of an user interface

Although the process issues are not the primary focus of this paper, but the modelling method, some considerations for the design process are presented here briefly.

We propose that designer approaches these tasks via design issues, which are put in form of questions. While carrying out the tasks, designer seeks answers to questions and draws related diagrams, which are utilized in software implementation phase, in base of the answers.

We suggest an incremental iterative design process, along the lines of the classic spiral model (Boehm 1988), i.e. not trying to get the design right at once but to improve diagrams, adaptors and shadows gradually. The more system requirements are taken into account in design diagrams, the need of change in adaptors' properties and shadows' rules is more apparent. To maintain the integrity of system design it is advisable to make the required adjustments right away to adaptors and shadows. Making it afterwards may break the design scheme due to shadows reusability.

Adaptive system is multidimensional and often a horrible mess. Getting a clear picture of a system being design is difficult and the risk of misinterpretation is high. The separation of adaptors and heuristics (shadows) from the rest of the system, and the ability to reuse shadows makes system design clearer, but despite of all this produced diagrams are complex. To our mind designer's mental burden should be ease by choosing a flexible design strategy and by using special design tools, which allows designer to examine the system in different viewpoints (in point of adaptors).

Modelling adaptivity

An easy way to understand the structural adaptation method, is to approach it through user adaptation. That is why the following presentation of adaptivity modelling is user centric, not because it is the only

adaptivity type supported. Furthermore, we believe that this approach deepens user modelling by forcing an user model more integral part of a system, instead of being attached on top of it.

In the following you will see partial design process of an adaptive system. The goal is to design system which contains scientific articles (publications) around our research project OWLA. These publications should be available, in different degrees, for system's users.

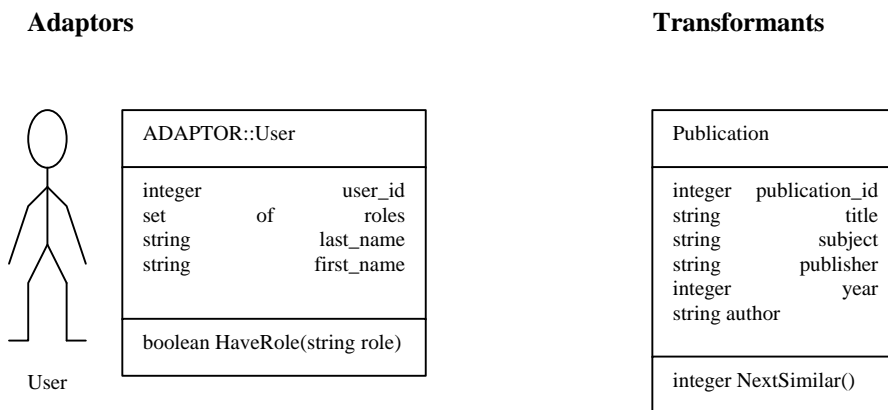


Figure 2. An adaptor and a transformant skeleton.

Modelling of an adaptive system should begin based on requirements gathering results – the overall view. The first design task is to identify and define preliminary adaptors and transformants, or adjust class definitions if such already exists.

Two objects were identified in requirements gathering: Publication and User. We start by creating corresponding classes "User" and "Publication". One of the requirements was that system adapts different users. Because of this "User" is defined as an adaptor class and "Publication" is left as a normal class (transformant). See figure 2.

In the requirements gathering we found out that system has three different users: Financiers who provides research capital, Researchers who are interested in research results (publications), and Press which is interested in new brilliant ideas. Each of these groups has own interests and demands.

Financiers require condensed and focused information. They need to know what they are paying for and how they can gain advantage of research results.

Researches should have full access to publications but because financiers are involved, we will have to limit the access rights.

The press need to know about what is being done (in general) and about new findings, if this does not conflict with interests of financiers.

We can now see that user's role (Financier, Researcher, or Press) is the dominant factor in adaptation - according to user's role, system produces different output.

This means that each publication should have own summary for each of the role: one summary for Financiers, one summary for Researchers, and one summary for Press. This also means that these summaries are Publication's class attributes - transformant attributes.

In base of this knowledge we can define shadows (see figure 3) and sketch Publication class with new attributes.

Shadows

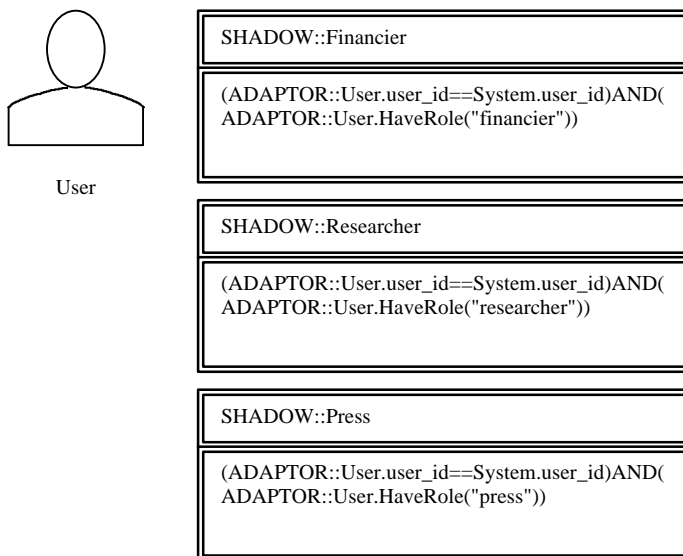


Figure 3. Shadows.

Limiting Researchers' access rights is still an open question. The policy is that researchers who are involved with the OWLA research project have right to view the related publication, other researchers may only view the summary.

We decided to secure publications by encoding them and passing the decoding key to researchers' who are qualified viewing whole publications.

We use User's class attribute "securityclass" to indicate qualification. The decoding key is also User's class attribute but it is a transformant because it is possessed only by certain group of users.

To accomplish the access restrictions we add the two new attributes to User class and define a new shadow. After this we can draw new User class where security issues have been taken account. See figures 4 and 5.

Shadows

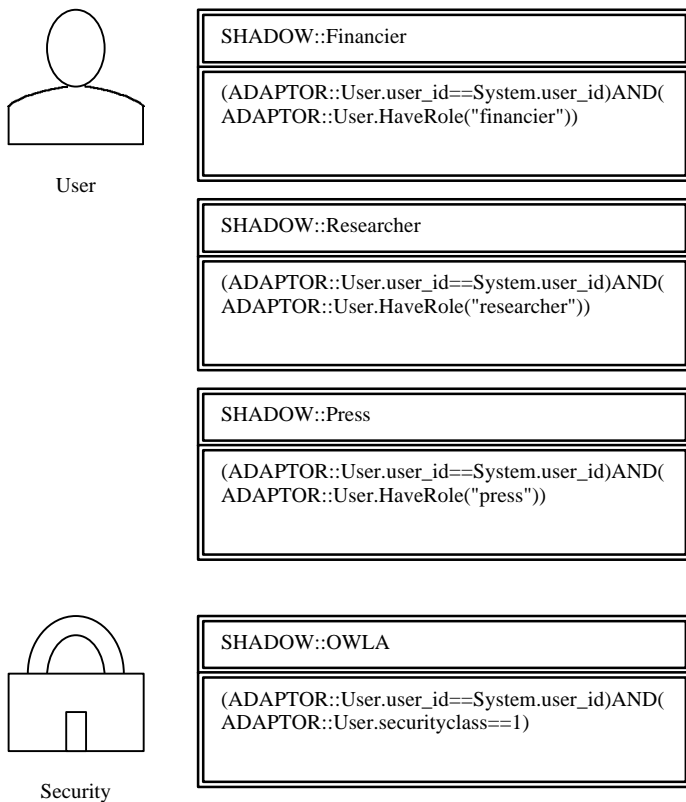


Figure 4. Completed shadows.

In the figure above you can see completed shadows, three user shadows and one security shadow, and in figure 5 is completed adaptors, one user adaptor with transformant.

Adaptors

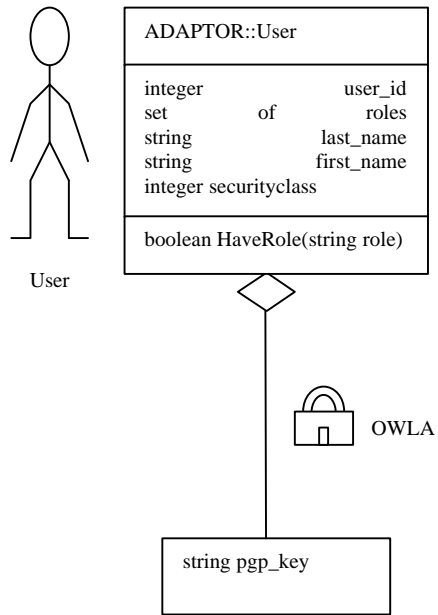


Figure 5. Completed adaptor.

We have now all the material we need to draw an Adapted Information Content Diagram (AIC) which is used for describing application's data structure and related functions. The previously defined Publication class and shadows are used. See figure 6.

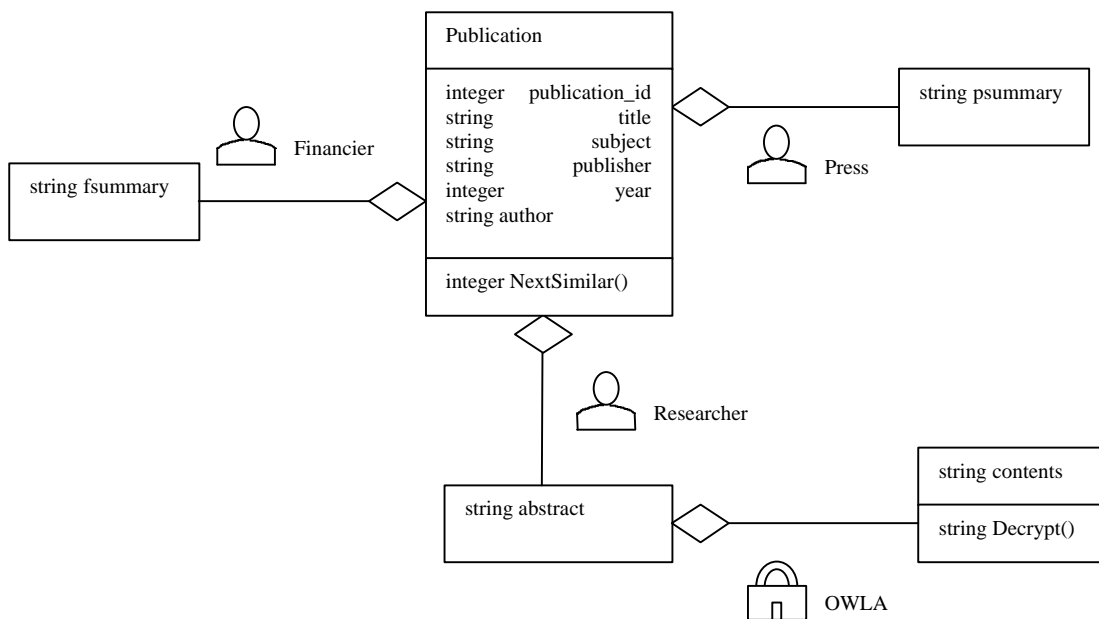


Figure 6. Adapted Information Content (AIC) -diagram.

Diagram's skeleton consists of one class, Publication, which has six attributes and one method. Completing the diagram there are four transformants connected to skeleton through four shadows. Each transformant has one attribute and one of them has also one method.

When you start reading adapted diagram please keep in mind that diagram's skeleton is formed by classes and transformants are their conditional extensions containing attributes and/or methods.

Reading the AIC-diagram is quite straightforward operation: you examine the from the adaptors' point of view, adaptor by adaptor if necessary. Skeleton is the part which is always there despite of adaptation. Transformants are adaptive parts which extend skeleton if adaptor has certain property values (what these values should be is determined by shadows).

Let us examine the AIC-diagram in point of User's view. First we assume that user has role of financier (User class attribute "roles" has value "financier").

Publication has six attributes (publication_id, title, subject, publisher, year, and author) and one method (NextSimilar) as you can see from the figure 6. In addition to this, because user has role financier, Publication class has also an extra attribute "fsummary". The reason why this attribute is included in Publication class and nothing else lies inside the shadows. Only the condition inside the Financier shadow is true, all other conditions inside the other shadows are false. See figure 4.

If user's role is changed to press, publication has the same six attributes and one method as in financier's case. The only difference is that instead of attribute "fsummary" there is an attribute "psummary". The

reason is shadow Press, which is the only shadow who condition is true when User class attribute "roles" has value "press".

Examining the rest of the diagram requires giving values to two attributes in User class. First attribute "roles" receive value "researcher", stating that user is a researcher, and then attribute "securityclass" is set to 1 indicating that this researcher is part of the OWLA team.

When we look at the figure 5 we can see that user is entitled to decryption key (attribute "pgp_key" belongs to User class) if OWLA shadow allows it, which it does if User class attribute "securityclass" has value 1.

Thus the Publication object appears to OWLA researcher as follows: Publication has the same six attributes (publication_id, title, subject, publisher, year, and author) and one method (NextSimilar) as in previous cases. Because User class attribute "roles" has a value "researcher", publication object has an extra attribute "abstract". Now when User class attribute "securityclass" was set to indicate that researcher is part of the OWLA team, publication has also attribute "contents" and method "Decrypt". In total OWLA researcher's Publication object has eight attributes and two methods.

This example is implemented on-line for the interested to try at <http://owla.oulu.fi/demo/publication.py>

Summary and open questions

We have developed an approach for designing adaptive hypermedia, and based on that a method for visualizing dependencies between the *adaptors* (i.e. objects respect to what the system adapts) and *transformants* (i.e. the parts that adapt) using so-called *shadows*, which hide the underlying algorithms (heuristics) and are meant to be an easy and intuitive tool for designers.

The method is intended to complement existing hypermedia design methodologies, such as OOHDM and AHDM, which currently lack means for showing the dependencies between e.g. content and properties in the user model in the design specifications for an adaptive hypermedia system. We have not yet, however, evaluated the proposed method in practice nor fully analyzed the questions related to fitting it in the existing design methodologies. For example, the use of shadows (illustrating adaptivity dependencies) in the various different design diagrams remains an open question (e.g. in OOHDM, could shadows be used in every diagram, or only in the navigational class diagram? How does adaptivity on lower levels affect the higher?).

There are also other types of adaptivity that this method does not currently address – in the words of our preliminary classification, it now deals with *adapting to difference* but ignores *adapting to change*. Addressing the questions related to this are left for future work.

We have recognized the need for tool support for the method – especially the possibility for the designer to examine the system with different views is an interesting function. Otherwise the concept of association class in UML (Fowler 1997, p. 93) can be used to model the shadows (as stereotypes) with standard tools, as we will be experimenting in real-world projects in near future.

References

Baumeister H., Koch N. and Mandel L. (1999). Towards a UML Extension for Hypermedia Design. In UML'99 The Unified Modeling Language - Beyond the Standard, LNCS 1723. Springer Verlag.

Boehm, B. (1988). A spiral model for software development and enhancement, *Computer* May, 61-72.

De Bra, P., Eklund, J., Kobsa, A., Brusilovsky, P., Hall, W. (1999). Adaptive Hypermedia: Purpose, Methods, and Techniques. The 10th ACM Conference on Hypertext and Hypermedia, Hypertext '99.

De Bra, P., Houben, G. J., Wu, H. (1999). AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. The 10th ACM Conference on Hypertext and Hypermedia, Hypertext '99.

Brusilovsky, P. (1996). Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Vol. 6, pp. 87-129, Kluwer academic publishers.

Fowler (1997). UML Distilled – Applying the standard object modeling language. Addison-Wesley Object Technology Series.

Koch, N. (1998). Towards a Methodology for Adaptive Hypermedia Systems Development. In *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen, Proceedings of the 6th Workshop, ABIS-98*, U. Timm and M. Roessel (Eds.)

Gómez J., Cachero C. and Pastor O. (2000). Extending a Conceptual Modelling Approach to Web Application Design. B. Wangler, L. Bergman (Eds.): CAiSE 2000, LNCS 1789, pp. 79-93. Springer-Verlag Berlin Heidelberg.

Schwabe, D. & Rossi, G. (1998). Developing Hypermedia Applications using OOHD. Workshop on Hypermedia Development Processes: Methods and Models, Hypertext'98. <http://www.inf.puc-rio.br/~schwabe/papers/ExOOHDM.pdf.gz>

Lieberherr, K. (1996) Adaptive Object-Oriented Software: The Demeter Method. PWS Publishing Company