Unleashing adaptivity for non-technical authors

Studies in Communication Sciences 5/1 (2005) 167-182

JACOPO ARMANI*

UNLEASHING ADAPTIVITY FOR NON-TECHNICAL AUTHORS.
A USER-CENTERED DESIGN LANGUAGE FOR EDUCATIONAL ADAPTIVE WEBSITES

Adaptive technologies have proven their effectiveness only in small-scale lab courses, thus they still wait for being released to the large community of practitioners. Among the causes of this, there is the difficult task of designing an interactive adaptive application, especially for non-technical groups of teachers and instructional designers. To solve this issue, we defined a design modeling language tailored to non-technical people. The language features a model-driven approach that allows it to be automatically implemented in a running application. The language supports ECA rules to represent adaptive decisions, a more natural way for non programmers. The application elements that can be adapted are: pages, fragments, and links. The types of adaptations that can be performed are limited to the most basic and reusable techniques only. Some results from a series of evaluations are available. Evaluations show that this set of adaptive techniques is complete enough to support several different application scenarios.

Keywords: educational adaptive hypermedia systems, design modeling language, ECA rules, conditional fragments, adaptive navigation support.

*University of Lugano, jacopo.armani@lu.unisi.ch

1. Introduction

The field of Adaptive Hypermedia Systems is a well established research area concerning all those, mainly web-based, applications that adapt themselves to the user's traits, preferences, context, and goals. Researches on AHS have been extensively tailored to different fields ranging from education to e-commerce.

1.1. Are Educators Using Educational Adaptive Hypermedia?

In the education area, although the adaptive technologies and architectures are well established (Brusilovksy 2001), still very few products have been successfully deployed.

There are some evidences, which suggest that adaptivity is having a hard time finding its way in the education context. As a matter of fact, if we take a look at the industry of e-learning packages (namely *learning management systems*, or LMS), which indeed are a good mirror of what are the current expectations and requirements of the education community, we notice that practically no systems support any design feature that can be considered *adaptive*. At most, a few of them have begun including some personalization features for the learner, providing a basic form of adaptability.

From a punctual analysis[1] we notice that just a handful of packages support some customization features for the student (i.e. ATutor, Desire2Learn, WebCT Vista). On the other side, all of the learning packages support content and structure customization for the teacher. Of course the reason is clear, since LMS are tailored to the community of educators, they must provide ways to let instructors create and modify their courses. Surprisingly, as we will see later on in this paper, this very characteristic is not supported by most of the EAHS developed so far.

Finally, if we cross a list of popular educational packages with the set of *state of the art* adaptive educational hypermedia systems (Brusilovsky 2001: 91; Brusilovsky & Peylo 2003: 160), we cannot find any overlapping element.

Thus we must conclude that the educators' community does not consider adaptive educational hypermedia systems as a practical resource yet.

[1] Source: EduTools (www.edutools.info), last checked on September 2004.

1.2. Why Educators Do Not Exploit Adaptivity?

Generally speaking, among the reasons, educational adaptive hypermedia systems (EHAS) are simply too difficult to design, set-up, and implement, due to the high technical competencies they require (Brusilovsky, 2003).

In particular, AHS seem to fail in providing practitioners (i.e. educators, instructional designers, teachers...) with sound and immediate benefits from using them. Too few empirical studies and contradictory results do not promote properly the benefits of applying such technologies to e-learning.

On the other hand, the costs to introduce AHS in real contexts are extremely high. In fact, usually the most common reasons to explain why such systems "are not used in practice are:

1. The *not-made-here reaction*, the feeling that the package does not fit local circumstances;
2. Lack of time and facilities for the instructor to integrate the package into a sustainable use;
3. Costs and problems with acquisition, maintenance, and updating" (Collis 2002: 7).

In fact, most of the available EAHS in literature are hard-wired applications, which cannot be exploited in other contexts but the designated one. These applications are one-shot design artifacts that evidently lack of reusability.

Recognizing this as a hindering issue, some researchers investigated some solutions. First of all, domain-independent EAHS were produced, such as ELM-ART (Weber & Möllenberg 1995), Interbook (Brusilovsky et al. 1998), KBS Hyperbook (Henze et al. 1999), AdLearn (Armani 2001).

Yet these systems required high competencies to be set-up because of the complex adaptive techniques they used. Therefore nowadays it is clear that the lack of authoring tools is one of the reasons for the low diffusion of adaptivity among educators.

In our opinion this analysis is correct but incomplete. In fact we claim that the field of EAHS is missing the focus on a more author-centered approach to the design of educational applications, such approach could help out our artifacts to achieve a real breakthrough in the community of educators, as many (non adaptive) LMS have obtained.

2. User-Centered Design of an EAHS

By placing the user at the center of the design process we mean gathering his goals and translating them into design solutions. We believe that

this approach may produce a new generation of adaptive systems that can be successfully adopted in practical education contexts.

From preliminary interviews with educators and literature review we formulated two strong hypotheses that would be beneficial in the creation of the right models and tools:

(i) every education package must be *reusable* (reusability)

(ii) teachers want to have *control* over the results of adaptive decisions (control).

Reusability of educational packages goes in the direction of reducing the hindering factors highlighted by Collis. In particular the *not-made-here reaction* can be addressed by designing general purposes systems.

Control is related to fear factors reported in some interviews with educators: artificial intelligence in fact seems to scare the common user. In particular authors mistrust the abuse of intelligent systems, whenever they cannot track nor predict the effects of automatic decisions made by the system.

In our vision, these two dimensions together shape a well defined design solution space, which include a design modeling language and an authoring environment that together allow to accomplish the author's goals.

Therefore we designed first a design modeling language tailored to non technical people, and then we implemented an AHS to execute its schemas. Moreover we also realized an authoring tool to visually design an adaptive website using the language.

In this paper we are focusing on the development of the design language, showing how it is possible to integrate existing adaptive technologies in a framework tailored to authors with little or no technical background.

3. Definition of a Design Modeling Language for Adaptivity

3.1. Setting the Scope of the Design Modeling Language

As already envisioned by some researchers, domain-independence is a *must-have* to enable educators adopt a system, therefore it is a preliminary requirement of our solution, yet it is not a sufficient condition. We need also a well designed modeling language to enable authors to set the specifications of their adaptive applications. Modeling languages differ in terms of scope and applicability. There are conceptual languages (such as HDM, Garzotto et al. 1993), and model-driven languages (for example WebML, Ceri et al. 2000). The former group represents those languages that are not bound to any implementation. Moreover they are also highly abstract,

delivering the authors from the details of a system language. The latter group instead aims at providing a modeling language that can be automatically implemented into a running application. They usually require engineering capabilities to deal with implementation issues.

The applicability of these classes of languages depends on the context. For our purposes, since educators are usually not engineers nor hypermedia designers, we need to define a language eventually translatable in an implementation in an automatic way. On the other hand our modeling language must be understandable by non-technical people, i.e. educators. Therefore we need to hybridize the two approaches.

3.2. Defining the Dictionary of Adaptive Techniques

The context of adaptive system is different from that of hypermedia design, since no standard adaptive operators, or structures have been identified yet. Therefore the definition of a design language is strictly intertwined with the definition of the supported adaptive techniques.

To set the boundaries of our design framework we have taken into account some application scenarios, which our solution should support (cf. Table I).

Table I: Application Scenarios for an EAHS

	Context				Instructor's traits	
	Teaching Environment	Domain structure	Design approach	Instructional Strategy	Motivation Resources	Computer Background
1	Online	Simple	Strategy driven	Lecture-based Learning Styles	High	Low
2	Blended	Complex	Content driven	Lecture-based Student's progress	Low	Low
3	Mixed	Mixed	Mixed	Learner remediation	Mixed	Mixed
4	Blended	Complex	Strategy driven	Problem-solving, Scaffolding	High	High

It is important to notice that some scenarios are dealing with online-only learning contexts, while others are blended approaches where online activities support traditional face-to-face activities. This feature seems to affect other features such as instructor's motivation and the instructional strate-

gy. In an exclusively online context the instructor will be keener to spend time crafting the online activity; also her choice of the instructional strategy will be affected by the online nature of the course, for example adopting learner remediation strategies to detect possible learning problems.

Domain complexity is another important factor. We can expect some design problems deriving from excessive complexity of the domain. Some of these issues could be removed by carefully analyzing their nature and providing abstraction and generalization primitives. Some of these issues may depend on the interactions between the domain complexity and the instructor's traits: if domain complexity is high, while motivation and computer background are low, then we can expect that the authoring task will be overwhelming for the instructor.

Finally, instructional strategies are an important factor to understand what are the instructors' goals, and designing the necessary primitives to enable authors using them.

In literature we may find numerous adaptive techniques that have been proposed in the education context. Besides, given the heterogeneity of scenarios, our investigation was concerned about those techniques that are at the same time (i) learnable by a non-technical teacher, (ii) friendly for authoring, reusable to accomplish different instructional goals, and (iii) adequate to construct more complex adaptive techniques. We put all of the adaptive techniques from Brusilovsky's review (2001) into a matrix summarizing all the requirements into two basic axes: *simplicity*, and *applicability* of the technique. The results of the analysis are shown in Figure 1.

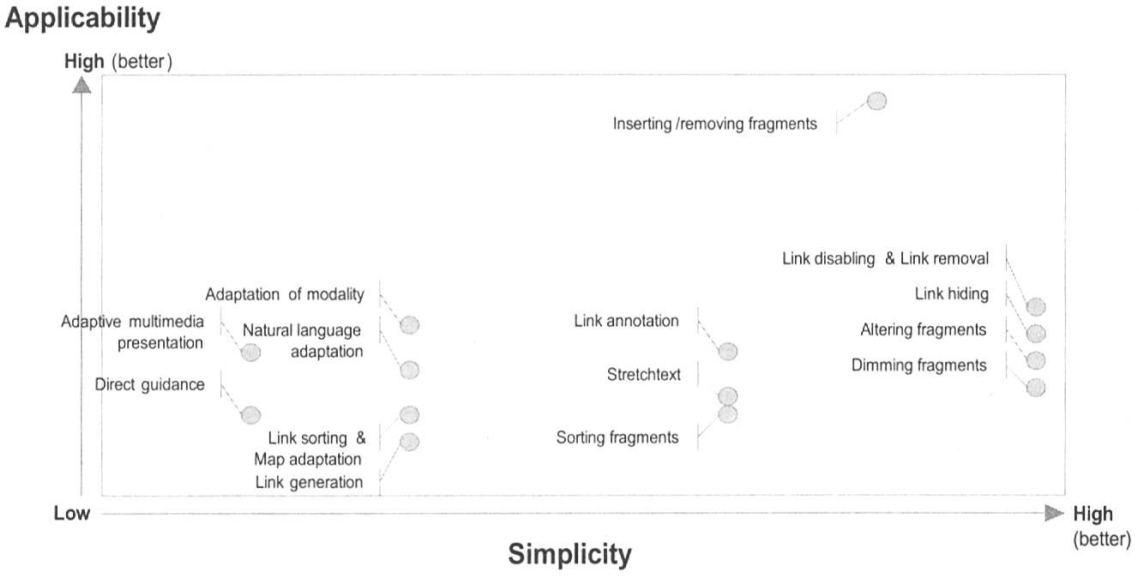

Fig. 1: Applicability and Simplicity Matrix of Adaptive Technologies

For our purposes, techniques with higher degrees of applicability (top of the diagram), and simplicity (right of the diagram) are preferable. This criterion leaded to the isolation of a cluster of techniques which likely constitute the answers to our requirements. These techniques are:

- Inserting/removing fragments: it is a fine grained technology, which is the basis for many higher-level adaptive techniques. Many implementations of it allow to insert or remove a fragment of content on the fly, based on information about the context, the user, and the session. Since the technique is a low-level brick, its implementation can be either complex, or simple, depending on the system's goals. It is worth noting that inserting and removing fragments can emulate many of the techniques of Brusilovksi's classification.
- Link hiding: this is a basic technique for hiding a hyperlink under certain conditions. Applying this technique the phenomenological nature of the link is removed, yet its functionality is still active. It can be used to implement direct guidance or as indirect navigation support. The technical skills necessary to set-up this technique may greatly vary from basic to demanding depending on the application.
- Link disabling: it is a complementary variant of the previous technique. It disables the link function of a hyperlink, while keeping intact the link appearance.
- Link removal: this technique removes the link from the screen. It can be simulated using an insert/removing fragment technique applied to a fragment containing a link only.
- Altering Fragments: it deals with the application of different styles (i.e. colors, fonts, etc...) to a text fragment. The very same considerations of the inserting/removing fragments technique apply in this case.
- Dimming Fragments: this is another technique to control the visibility of a text fragment depending on the context. With this technique a fragment can be dimmed or undimmed in order to modulate the user's reading attention. This technique may be simulated by the mean of altering the fragment style, or by a insert/removal fragment technique.
- Link annotation: these techniques permit to add text or icons to an hyperlink, commenting its nature, purpose, and relevance for the user.
- Stretch text: With these techniques a fragment of text can be presented with different lengths depending on context information. In some circumstances a short text description is given, while in other the full length version is shown. Anyway, in every moment the user can stretch down the short description to see the full-length version of the fragment and back.

3.3. Describing the Domain

Traditionally AHS need some information about the structure of the contents. This requirement leads to the definition of a domain model. This subcomponent of an AHS is usually very complex, providing the system with metadata about the domain entities (concepts, pages, etc.) and relationships (semantic networks, navigational schemas, etc.), along with other relevant information (media types, keywords, etc.). However, evidently we cannot ask a teacher to create and keep updated complex data structures, just to meet the system's expectations.
Therefore we envisage a simplified domain model which includes:

- a set of content nodes (*resource*), which are the virtual locations where the actual content is displayed to the learners
- basic structural hierarchical relations between each node
- navigational hyperlinks between each node to allow horizontal and semantic navigation.

This model is a oversimplified version of a hypermedia conceptual model, such as HDM, but it grasps all of the hypermedia elements that are both understandable and expressive enough to define a web-based course.

3.4. Describing the Context

Adaptivity is based on knowledge of the context of use, therefore our modeling language must provide primitives to describe it. In our vision we distinguish the context into two components: the user, and the session.

User Profile
The user, as a single person, has many traits that can be described to the system, hence modeled, for adaptive purposes. The list of relevant traits largely depends on the interaction and communicative goals. For example to provide lessons tailored to the user's learning styles we must track his/her preferences. If we want to progressively disclose contents of increasing complexity, we must store information about the user's browsing and learning history.

Therefore the user profile must be an extendable and customizable entity, with a set of operators to add new information (ex. Updating the user's browsing history), and retrieve information previously stored.

In the field there are many proposals for user profile and models adopting machine learning techniques (ex. Bayesian or neural networks, data mining servers, etc.) which are far too complex for our intended audience. Therefore we will stick just to straightforward user modeling techniques, such as vector-based knowledge bases, which enable the authors to represent most of the relevant user's traits with ease.

Session Profile
In addition to information about the user, we want also to provide information about the situation, or session which the user is browsing in. Information about the session are, for instance, the current time, the day of the week, the date, during which the interaction occurs. Moreover information about the amount of bandwidth the users have at their disposition can be part of the session profile as well.

3.5. Defining the Language Entities and Operators

The selection of adaptive techniques we made earlier permits to identify the necessary entities that our modeling language must include: along with the domain model, and the context model elements, evidently fragments and links are the basic referents that are required.

Since links are largely targeted by adaptive techniques (i.e. link hiding, or link annotation techniques), we envisage some abstraction primitives in our language to save time during the modeling of the course. For this reason our language supports link adaptation at the level of classes of links. A link is a relation between a source page and a target page. Therefore it is possible to distinguish several classes of links:

- The class of links going from A to B[2];
- The class of links going from <everywhere> to B;
- The class of links going from A to <everywhere>.

This distinction of classes of links is very useful when we must write adaptation rules, because it allows to state rules that apply to every member of a given class with a single statement. This economy of rules is very important when it comes to working with hundred (possibly thousands) of small objects such the hyperlinks.

[2] We refer to the links going from A to B as a class of links, because the actual page may show several instances of this relation dislocated around the page, therefore it is correct to call them a *class.*

3.6. Adaptation Rule Language

The adaptation behavior of the systems is expressed by the mean of rules exploiting the ECA (Event-Condition-Action) paradigm. This powerful standard approach enables authors to locally define events and conditions that trigger the execution of an action. Events can be drawn from a list of relevant checkpoints during the user/system interaction flow: for instance, an event occurs when (i) the user has logged in or out, or (ii) has accessed a resource, or when (iii)the system has refused access to a resource.

Conditions are Boolean expressions on the status of the domain, context, and user models. For example valid conditions are:

(i) The current date is November 1st 2004

(ii) The user's language is German

(iii) The user's seen Resource A and the user has not seen yet Resource B.

The actions modify the state of some elements of the models (*Tracking Rules*), or of some presentation attributes of resources, fragments, and links (*Presentation Rules*). A complete list of all the available actions is shown in Table II, and Table III.

Table II: List of Presentation Rules

Target Type	Action	Description
Resource rules	Grant	Grant access to the resource
	Deny	Deny access to the r esource
	Redirect	Redirect the user to another resource
Fragment rules	Hide	Remove the fragment from the page
	Show	Show the fragment in the pa ge
	Alter (font, color, background, border)	Alter one or more fragment attributes such as the font, the colors, etc…
	Stretch (length, open/close)	Set a stretch -text for the given fragment. When closed the fragment will show the number of characters spec ified in length. The stretch -text may be set open or closed by default.
Link rules	Hide	Remove the link from the page
	Disable	Remove the anchor (but the link it is still visible), thus disabling it
	Style(style)	Apply a specific style to the link (Ex. Color, font, and so forth…)
	Add a note(text)	Append a pop -up note to the link
	Add an icon(icon)	Append an icon to the link

Table III: List of Tracking Rules

Action	Description
Set value	Set a value for variable
Sum	Sum a value to a variable
Subtract	Subtract a value from a variable
Concatenate	Concatenate two strings
Flip	Flip the value of a Boolean variable

3.7. The Design Modeling Language in Action

An educator may create set of rules in the form of condition-action statements to basically craft several different types of interactions within the website. For example, an instructor may be willing to set-up a self-learning module powered by an intelligent tutor. The tutor could be in charge of giving the student feedback about her progress in the module. In order to shape this particular interaction modality, the author may rely on a set of tracking rules to update student's progress. Then, specific presentation rule may be written to make the tutor seeming intelligent. A tracking rule could be the following (in pseudo-code):

IF user accesses the page "Chapter 1"
THEN set the value of the variable "Ch_1_Started" equal to TRUE.

This rule basically contributes to the definition and update of the student profile.

Then, the tutor could be created by a list of presentation rules that, depending on the values stored in the student profile, present or not specific sentences. Each sentence may be implemented by the mean of removable fragments, that may appear or not on the web page. For instance, the tutor could be programmed to remind the student to finish the Chapter 1, once he has started it. This is the pseudo-code to build the rule:

IF "Ch_1_Started" Variable = TRUE
THEN Show Fragment ("You've started Chapter 1. I suggest you to keep reading it until the end. Please follow this *link* to go to the chapter")

Several of these rules would eventually shape a rich interaction modality between the user and the adaptive course.

It is worth noting that, although these examples have been presented in pseudo-code, an actual implementation of the language is available. This implementation adopts a human-readable system language based on XML. A working implementation of a system supporting this language has been discussed in (Armani 2004a; Armani 2005, Ch. 4). This system,

named ADLEGO[3], includes most of the elements we defined in our design language but exploiting a system-oriented syntax. Although this approach is computationally efficient, it does not foster user-friendliness. For this reason, we also designed an authoring tool to visually present the design schemas and rules to the authors. In this way, educators may be able to visually manipulate language primitives in a more natural manner (Armani 2004b; Armani 2005, Ch. 5). With this authoring tool the educators may create all the necessary rules just by dragging and dropping elements such pages, links, and variables to form rule conditions and actions. The tool then translates the adaptation rules into the system syntax.

4. Evaluation of the Design Framework

Usability tests performed in a small-scale lab experiment have shown that the design language is easy enough to be grasped by non-technical teachers (cf. Armani 2005, Ch. 6). The testers were all instructors with very basic experience in IT. Their goal was to create an adaptive course of a small-medium size (around fifty pages, or in student's terms up to ten hours of work.

The evaluation framework was scenario-based. Each instructor was given a set of tasks to perform, including creating complex rules to achieve a specific interaction goal.

The number of requests for assistance, and backtracks were tracked to measure the effectiveness of the authoring environment.

Although the results of this experiment were mainly related to the authoring tool, they evidently reflected the effectiveness of underlying language primitives we illustrated in this paper. Picking up too complex adaptive techniques would have hindered the usability of the language and of the authoring interface as well.

A more focused evaluation on the language expressiveness was performed by exploiting it to create a couple of adaptive modules on Cognitive Psychology. Each module implemented specific personalization strategies. During this evaluation the language proved to be powerful enough to design complex interactions. For instance, the instructor was able to create an adaptive tutor who was in charge of leading the students through a set of contents and virtual experiments exploiting only the show/hide fragment rules (cf. Figure 2). Link annotation (*add icon*, and *add text* rules) and link disabling techniques refined the learner's experience by providing tailored explanations and suggestions during navigation.

[3] The ADLEGO system is available for download at: http://www.istituti.usilu.net/armanij

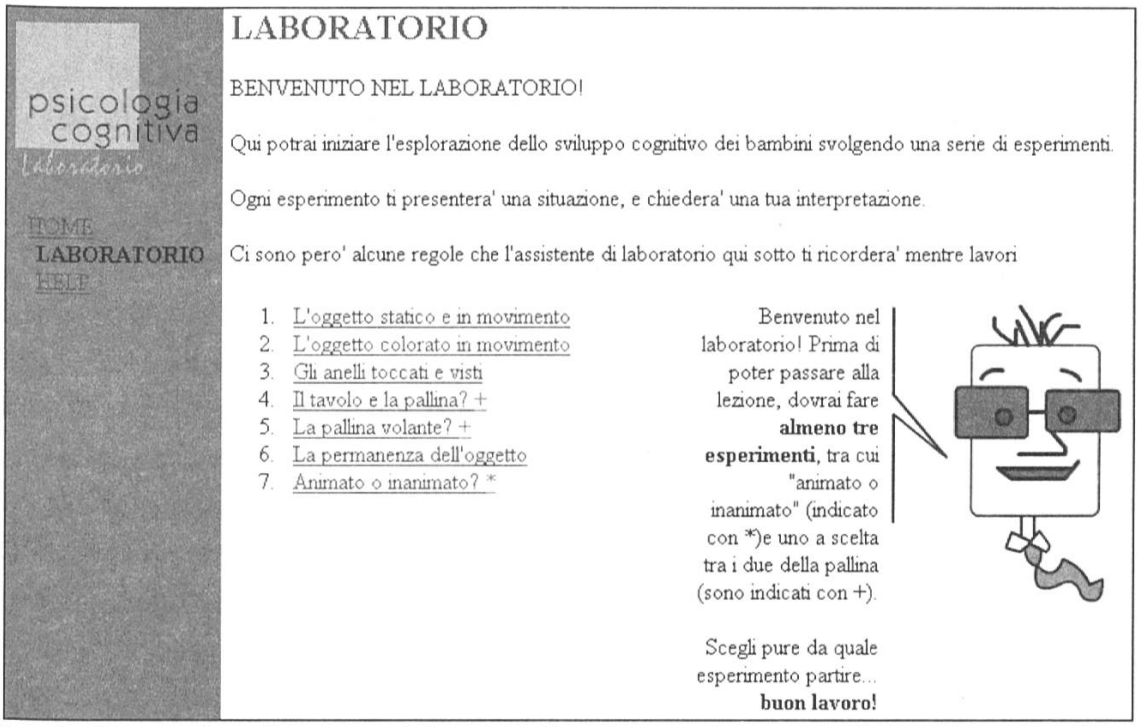

Fig. 2: Screenshot from an Adaptive Course on Cognitive Psychology developed with the Modeling Language

The only source of troubles we isolated was related to the use of Boolean conditions in connection with negative actions, like hiding a text fragment unless a certain condition is met. In these circumstances the testers were easily tricked into writing double negations statements. These statements of course expressed the opposite meaning they wanted to state. For instance, when a user wanted to state: "Do not show the link to *Page2* until the user has not read *Page1*", they often ended up writing a rule like the following:

IF "Page1_visited" variable is FALSE
THEN Show link to Page 2.

These kinds of mistake are more likely related to the misinterpretation of Boolean operators, than to poor language design. Besides, being aware of them may help us selecting a isolating a more reliable selection of Boolean operators.

5. Related Works

As already anticipated, some researchers have tried to develop design languages for adaptivity. For instance, the Adaptive Hypermedia Development Methodology (Koch 1998) supports adaptive hypermedia

by including the design of user models within the context of a standard hypermedia design methodology. The methodology is based on an object-oriented modeling approach. The main differences with our approach are that AHDM adopts UML as its syntactical device, and its software engineering nature. These features have a two-fold consequence: firstly, they reduce language usability for a non-technical author, and moreover they do not allow an automatic implementation of the produced schemas into a running application.

Other design methods have been empirically inferred from direct experiences on the creation of adaptive systems with teachers and instructional designers (Calvi & Cristea 2002; Allert et al. 2002). A good example of this approach is MOT (Cristea & Calvi 2003), an adaptive system implementing a three-layered specification language for adaptive techniques tailored to the educational context. MOT language includes high-level primitives that can be used by an author to describe the semantics of the application, leaving out the technicalities of its implementation. Its layered approach allows an author to concentrate on different levels of granularity, from a goal-oriented (high-level adaptation layer, or adaptation strategies layer), through a more domain-oriented (medium level, or adaptation language layer), down to a system-dependent (lowest level, or direct adaptation techniques layer) level of design. The main difference with our approach is that the high-level layer of adaptation is automatically translated by the system in direct adaptation techniques, thus reducing the author's control over the results of such adaptations. On the contrary we envisage to keep adaptivity under control by allowing the authors to directly manipulate adaptation techniques only. In our framework the necessary generalizations can be obtained by the mean of design patterns that integrate adaptation techniques together to form recurrent higher-level behaviors.

Other related design languages are MAID (Armani & Botturi 2003) and LAOS (Cristea & De Mooij 2003). Yet their approaches support a high-level description of the adaptive interactions, leaving a gap between the specification schema that they deliver, and the necessary implementation decisions that must be made.

6. Conclusions

A user-centered modeling language for the design of educational adaptive websites has been presented. This work is part of a larger effort that aims at diffusing adaptive technologies in the community of practices.

The present design language features a low-level approach giving the author full control over the adaptive techniques and their results. Its nature allows also an straightforward implementation of the produced blueprints into a running adaptive application.

Evaluations of the language expressiveness and usability show promising results. The language was successfully exploited to produce some adaptive courses using simple primitives to build complex interactions.
From these experiences we noticed that some abstraction capabilities are needed in order to ensure economy of design. We claim that these capabilities must be pursuit by the mean of design patterns that aggregate successful applications of low-level adaptive techniques. This approach should be preferred to a simple extension of the language with more abstract features. In fact, in this way we keep the controllability of the results of adaptation in the author's hands, and hence we do not delegate the author's responsibility to the application.

Acknowledgments

The current work has been partially carried out under the supervision of Prof. Brusilovsky, School of Information Sciences, University of Pittsburgh, PA.

References

ALLERT, H.; RICHTER, C. & NEJDL, W. (2002). Learning objects and the semantic web - explicitly modelling instructional theories and paradigms. Proceedings of E-Learn 2002: WorldConference on E-Learning in Corporate, Government, Healthcare, & Higher Education, Montreal. Norfolk, VA: AACE Press.

ARMANI, J. (2004a). Shaping Learning Adaptive Technologies for Teachers: a Proposal for an Adaptive Learning Management System. Proceedings of ICALT 04, Jonsuu.

ARMANI, J. (2004b). Visualizing Adaptivity for Teachers: an Authoring Tool for Designing Educational Adaptive Websites. Proceedings of the Authoring Adaptable and Adaptive Educational Hyermediaworkshop, AH 2004 Conference, August 27th – 29th 2004, Eindhoven.

ARMANI, J. (2005). Taming Adaptive Technologies for Education. Ph.D. Thesis. Lugano, University Library.

ARMANI, J. & BOTTURI, L. (2003). Adaptive Hypermedia System Design: a Method from Practice. Proc. of WWW/Internet 2003 International Conference, Algarve.

BRUSILOVSKY, P. (2001). Adaptive hypermedia. User Modeling and User Adapted Interaction. *Ten Year Anniversary Issue* 11/1-2: 87-110.

BRUSILOVSKY, P. (2003). Developing adaptive educational hypermedia systems: From design models to authoring tools. In: MURRAY, T.; BLESSING, S. & AINSWORTH S. (eds.). Authoring Tools for Advanced Technology Learning Environment. Dordrecht: Kluwer Academic Publishers: 377-409.

BRUSILOVSKY, P. & PEYLO, C. (2003). Adaptive and intelligent Web-based educational systems. *International Journal of Artificial Intelligence in Education, Special Issue on Adaptive and Intelligent Web-based Educational Systems* 13/2-4: 159-172.

BRUSILOVSKY, P.; EKLUND, J. & SCHWARZ, E. (1998). Web-based education for all: A tool for developing adaptive courseware. Computer Networks and ISDN Systems. Proceedings of Seventh International World Wide Web Conference, 30 /1-7: 291-300.

CERI, S.; FRATERNALI, P. & BONGIO, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. *WWW9/Computer Networks* 33/1-6: 137-157.

COLLIS, B. (2002). Information Technologies for Education and Training. In: ADESLBERGER, H.H.; COLLIS, B. & PAWLOWSKI, J. M. (eds.). Handbook on Information Technologies for Education and Training. Berlin: Springer-Verlag: 1-20.

CRISTEA, A.I. & CALVI, L. (2003). The three Layers of Adaptation Granularity. In: BRUSILOVSKY, P.; CORBETT, A. & de ROSIS, F. (eds.). User Modeling 2003. 9th Proceedings of the International Conference, UM 2003 Johnstown, PA, June 22th-26th. Heidelberg: Springer Verlag: 4-14.

CRISTEA, A.I. & DE MOOIJ, A. (2003). LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. Proceedings of the 12th international conference on World Wide Web, 2003, Budapest. Alternate Track on Education.

GARZOTTO, F.; SCHWABE, D. & PAOLINI, P. (1993). HDM - A Model Based Approach to Hypermedia Application Design. *ACM Transactions on Information Systems* 11 /1: 1-26.

HENZE, N. & NEJDL, W. (2003). Logically characterizing adaptive educational hypermedia systems. Proceedings of AH 2003 Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems: 15-28. Retrieved at http://wwwis.win.tue.nl/ah2003/proceedings/numberedproceedings.pdf.

HENZE, N. et al. (1999). Adaptive hyperbooks for constructivist teaching. *Künstliche Intelligenz* 4: 26-31.

KOCH, N. (1998). Towards a Methodology for Adaptive Hypermedia Systems Development. In: TIMM, U. & ROESSEL, M. (eds.). Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen. Proceedings of the 6th Workshop, ABIS-98, Erlangen.

WEBER, G. & MOELLENBERG, A. (1995). ELM-Programming-Environment: A Tutoring System for LISP Beginners. In: WENDER, K. F. et al. (eds.). Cognition and Computer Programming. Norwood, NJ: Ablex: 373-408.