

VIDET: a Visual Authoring Tool for Adaptive Websites Tailored to Non-Programmer Teachers

Jacopo Armani

Institute of Communication Technologies
Università della Svizzera italiana
Via G. Buffi 13, 6900 Lugano, Switzerland
jacopo.armani@lu.unisi.ch
<http://www.istituti.usilu.net/armanij>

ABSTRACT

So far educational adaptive technologies have proven their effectiveness only in small-scale lab courses, thus they still wait for being released to the large community of educators. Among the reasons, there is the difficult task of designing and authoring an interactive adaptive course, especially for non-technical group of educators. In this paper we present VIDET, a visual authoring tool for designing web-based adaptive courses with the ADLEGO adaptive engine. The authoring tool is tailored to the community of non-technical instructors. It is pedagogic-neutral, allowing to define several different instructional strategies. The VIDET tool was implemented as a prototype in Macromedia Flash. Some results from a summative evaluation are available: the tool is effective at reducing the instructor's burden for creating structured adaptive courses. The basic set of adaptive techniques available is complete enough to support several different application scenarios.

Keywords

Educational adaptive hypermedia systems, Authoring tools, Visual programming, Web-based education, Macromedia Flash

Introduction

Adaptive technologies in the field of education have proven so far their effectiveness only in small lab experiments, thus they are still waiting for being presented to the large community of educators. First of all, as pointed out by some recent studies (Brusilovsky, 2003), educational adaptive hypermedia systems (EHAS) are difficult to design, set-up, and implement, due to the high technical competencies they require to master them. In particular, all of the (few) existing general purpose educational adaptive systems have a steep learning curve, that forbids a non technical teacher to autonomously create a course.

More generally speaking, the main issues that hinder the spread of many of the available adaptive systems in community of educators are:

1. High technical competencies to set up an adaptive course (i.e. writing of XML descriptors, textual configuration files, ...);
2. Difficulty in specifying in the system language the interactions that must occur between the user and the system (i.e. definition of concept networks, condition statements, resource indexing, ...);
3. Lack of ready-to-use patterns that exploit frequent adaptive teaching strategies.

Despite this situation, different researchers point out the importance of adaptivity in the definition of effective learning scenarios. For example, studies from the Instructional Design field (Park & Lee, 2004) show how much the adaptive instruction paradigm has been, and still is, a common trait in every day instructional situation: a teacher in a classroom naturally adapts his/her learning goal, presentation style, instructional strategy, and language to match the needs of his/her class, thus, why this cannot happen online too?

The Authoring Process of EAHS

Since an Educational Adaptive Hypermedia System is a hypermedia system, firstly we can refer to the well developed corpus of literature on Hypermedia Design to extract general features of the design process of hypermedia applications. In this field several conceptual models and methodologies have been developed. Just to mention a few: RMM (Isakowitz et al., 1995), OO-HDM (Schwabe et al., 1998), HDM (Garzotto et al., 1993), WebML (Ceri et al. 2000). Although adopting different and incompatible name conventions, all of them agree on the fact that essentially a hypermedia application is composed of (Coda et al., 1998): content, structure, access (or navigation), layout.

To the proposed list we can add a fifth element that is *context*, and we obtain a generic adaptive hypermedia model (as the AHAM model, cf. De Bra et al., 1999). Therefore each element of this list must be taken into account during the design process of virtually any AHS.

An interesting attempt of defining a general design model for AHS is the 5 layers LAOS model (Cristea and De Mooij, 2003) that includes the following elements: Domain Model, Goal Model, User Model, Adaptation Model, and Presentation Model. Interestingly the researchers also sketched a list of authoring tasks for any AHS:

1. Write concepts and concept hierarchy
2. Define concept attributes (define main and extra attributes)
3. Fill concept attributes (write contents)
4. Add content related adaptive features regarding GM (design alternatives – AND, OR, weights, etc.)
5. Add UM related features
6. Decide among adaptation strategies, write in adaptation language medium-level adaptation rules or give the complete set of low level rules (such as condition-action or IF-THEN rules).
7. Define format (presentation means-related; define chapters)
8. Add adaptive features regarding presentation means (define variable page lengths, variables for figure display, formats, synchronizations points, etc.).

The main benefit of this model is its generality, because it fits well with virtually any adaptive hypermedia system. Unfortunately it is too complex for a non-expert of the field, and it requires a deep understanding of each layer (and its sub-layers as well) to master it. Besides, it is too heavy for small educational contexts, thus scalability is a problem.

There are complementary approaches that aim at simplifying the authoring process: an example of this approach is the TANGOW's authoring model (Carro et al., 2000).

In TANGOW an author has at her disposition the basic concept of Teaching Task (TT) only. A Teaching Task can be atomic or composed, the latter being composed by other TTs. This way a simple hierarchy of TTs can be defined. Composed TTs can be presented with two sequencing strategies: a sequenced (*and*) or an alternative (*or*) strategy. The former imposes to complete all of the sub-tasks to complete the whole task, while the latter allows completing any subtask. Finally, TTs can be dependent upon completion of other TTs, therefore creating a dependency graph of Teaching Tasks. This authoring model is very simple and straightforward. Moreover it is based on a naïve, yet sound, understanding of basic Instructional Design methods (cf. E²ML, in Botturi 2003, as the closest example). On the other hand this system does not allow to creatively extend the adaptive behavior over the given few sequencing methods. Moreover its application is limited to the education field only.

Some insights on the theoretical authoring process of EAHS come from a recent study about authoring learning styles in adaptive hypermedia (Stash et al., 2004). In this research the authors experimented with authoring adaptive courseware using two different perspectives: an *adaptive engine pull*, and an *authoring push*. The former method deals with manually writing low-level adaptation rules to mimic the intended interaction with the user in order to achieve a specific instructional goal, namely in the example a learning style matching goal. The latter relates to choosing an instructional strategy among a dictionary of predefined strategies and letting the system define and perform automatically the necessary interactions with the user. The authors argue that "...it is important to study these two perspectives, as the one tells us what authors might want to see their educational adaptive hypermedia do, whereas the other one tells us what such systems can do at present." (*ibidem* p.122). The authors also report the importance of the granularity of the design method: some approaches work at the instance level of the course (namely single pages, and rules), whilst others are related to the schema level (general concepts). Both of them seem promising, yet from our experience it seems that non-technical people seem more keen reasoning in terms of instances (thus locally), than in general terms or at the schema level.

An important source of inspiration in the field is Brusilovsky's recent review on the authoring and design methods for AHS (Brusilovsky, 2003). In the author's opinion there are two stages to consider: *design* and *authoring*. Each stage involves several tasks as shown in Table 1.

With respect to the list of authoring tasks presented by (Cristea et al., 2003) we notice that this is a theoretical representation of the design process, while the other is an empirical one, which is derived from an actual methodology. Yet, both of them strive for the diffusion of a highly structured design method which contrasts with the expected end-users (non technical instructors) that should benefit from them. Therefore the best solution seems to be: pruning out some elements from these heavy methods, leaving only the most basic and relevant steps that may be helpful in practice for the large community of educators.

Table 1. Design and authoring steps in the process of creating adaptive hypermedia systems (from Brusilovsky 2003)

Stage	Task
Design	Design and structure the knowledge space Design a generic user model Design a set of learning goals Design and structure the hyperspace of educational material Design and structure the hyperspace of educational material Design connections between the knowledge space and the hyperspace of educational material
Authoring	Create page content Define links between pages Create some description of each knowledge element Define links between knowledge elements Define links between knowledge elements and pages with educational material

From our experience with the design and authoring of adaptive courses, we came across with similar results. In (Armani & Botturi, 2003) we presented a design method named MAID for designing adaptive courses with adaptive engines. MAID method concerns with the following steps:

1. Interaction Model: define the application behavior;
2. Domain Model: define the content structure;
3. User Model: define the relevant information about the user;
4. Interface Model: map the behavior into user interface elements;
5. Implement & Testing: implement and test the system.

With respect to the former methods, MAID takes into consideration all of the aspects of the design process ranging from requirement analysis to implementation and testing. Moreover it is a practical method that can be taught in a whole day session. Unfortunately, although the method is quite simple to learn, the tasks involved are still complex and often too structured to be executed properly by a non-expert without assistance from technical experts. Therefore an interesting question is whether we can prune some parts of these methods to reduce the most complex design and authoring tasks, without losing too much flexibility.

Authoring Tools for EAHS

The analysis of the authoring process of an EHAS is usually realized by the creation of an authoring tool to support it. Since the degree of maturity of the research on general principles for the design and authoring of EAHS is still low, we can count only a bunch of proposals for authoring tools.

In the last decade, some domain-independent Educational Adaptive Hypermedia Systems (EAHS), with different degrees of authoring capabilities were released, some important examples being: Interbook (Brusilovsky et al., 1998), Net-Coach (Weber et al., 2001), and AHA! (De Bra & Calvi, 1998). Yet, these tools have been used only within the research group who developed them (with very rare exceptions, for example the Author of this paper experimented with AHA! to design a set of adaptive courses to assess the usability of the tool, cf. Armani & Botturi, 2003).

More recently some research projects have directly addressed the authoring problem, by developing tools to support the process. Among the most relevant examples we can list: Schoolbook (Kupka et al., 2004), SIMQUEST (van Joolingen & de Jong 2003) REDEEM (Ainsworth et al., 2003), IRIS (Arruarte et al., 2003), Eon (Murray, 2003), and LEAP (Sparks et al., 2003). REDEEM, and Eon are the products most related to our framework, therefore we will present them briefly.

REDEEM shares with our solution the same goal: allowing teachers and instructors with little technical knowledge to create simple Intelligent Tutoring Systems. Although the goal is the same, REDEEM approach differs completely in the solution that it adopts. In fact, within the tool the instructor can customize her target audience in terms of categories (stereotypes), personalize the teaching strategies to use in the course, and create content sections and multiple choice exercises for learning assessment. REDEEM then is in charge of actually implementing the necessary logics to fulfill the teacher's specification. Therefore with REDEEM there are no ways to create new interactions that have not been foreseen by the designers. Moreover the teacher has no control over the translation of her specifications into user-system interactions.

Eon is a suite of domain independent tools for authoring all aspects of a knowledge based tutor, therefore the domain model, the teaching strategies, the student model, and the learning environment. The most interesting tools of the suite are the Flowline editor, which allows to graphically create teaching procedural strategies, and the ontology editor that allows to visually create a domain model of the application. Both of these tools aim at unleashing the control over adaptivity and interaction of the application to the instructor. Eon shares with our approach the objective of giving the teacher full control over the adaptive operations being performed.

Design patterns for EAHS

The idea of design patterns for adaptivity is not new (cf. Avgeriou et al., 2004; SCORM, 2004). Interestingly, the SCORM consortium recently suggested over twenty different kinds of learning patterns that have, at different degrees, some kind of adaptivity: from simple personalized learning environment, to advanced trial-and-error environments. Surprisingly the practice of design pattern did not start yet, even though it is doubtless that it would be extremely beneficial for educators and, in the end, for learners. One of the reasons is that educators have lacked of the right tools to create and manipulate patterns. We envisage that creating visually rich authoring environments may, in the end, open the opportunity to study and develop patterns of adaptivity for education. Until patterns are buried under heavy XML descriptions, no actual instructors would ever consider them as practical tools for their teaching goals. By the way, according to past studies (Finzer & Gould, 1993) the use of patterns proved to be a key feature for supporting educators in unleashing their creative and teaching skills with technologies.

Introduction to ADLEGO Adaptive Engine

ADLEGO is a low-level adaptive rule-based engine in the form of a web server plug-in that monitors the requests sent to the web server by clients through HTTP protocol.

ADLEGO Data Model

ADLEGO data model, shown in Figure 1, is based on the notion of *course*. The course is the container of all of the *resources*. A resource, in ADLEGO terminology, is a container of *sources* (or content sources). A content source can be: a *placeholder* which contains only a string of text as a reminder to be substituted at a later stage of the authoring process, or an *adaptive page* which is a basic html augmented with embedded rules to perform inclusion/removal of fragments, or an *URL to a page located in an external website*.

In addition a resource has a graphic template to present its content with a particular layout and look & feel. Finally, two sets of rules can be associated to a resource. The presentation rules affect graphic or navigation attributes of the resource, for example changing link colors, adding link icons, or denying access to a resource. A presentation rule structure is as follows:

- Target: attribute that is affected by the rule (ex. All links to resource “Introduction”, or Accessibility to this resource)
- Condition: a Boolean condition over values taken from context variables (ex. User’s traits, time variables, etc...)
- Action: an action performed on the target when the condition is true (ex. Add an icon to the link specified in the target).

Tracking rules instead are responsible for user model updates. Their structure looks like the presentation rule structure, but instead of targets there is an event source to specify under which circumstances the rule is triggered (ex. Click on a link, access to a page, user’s login...).

This information is stored in the *Course Description* file, which is accessed by the ADLEGO system every time a resource is requested by a client.

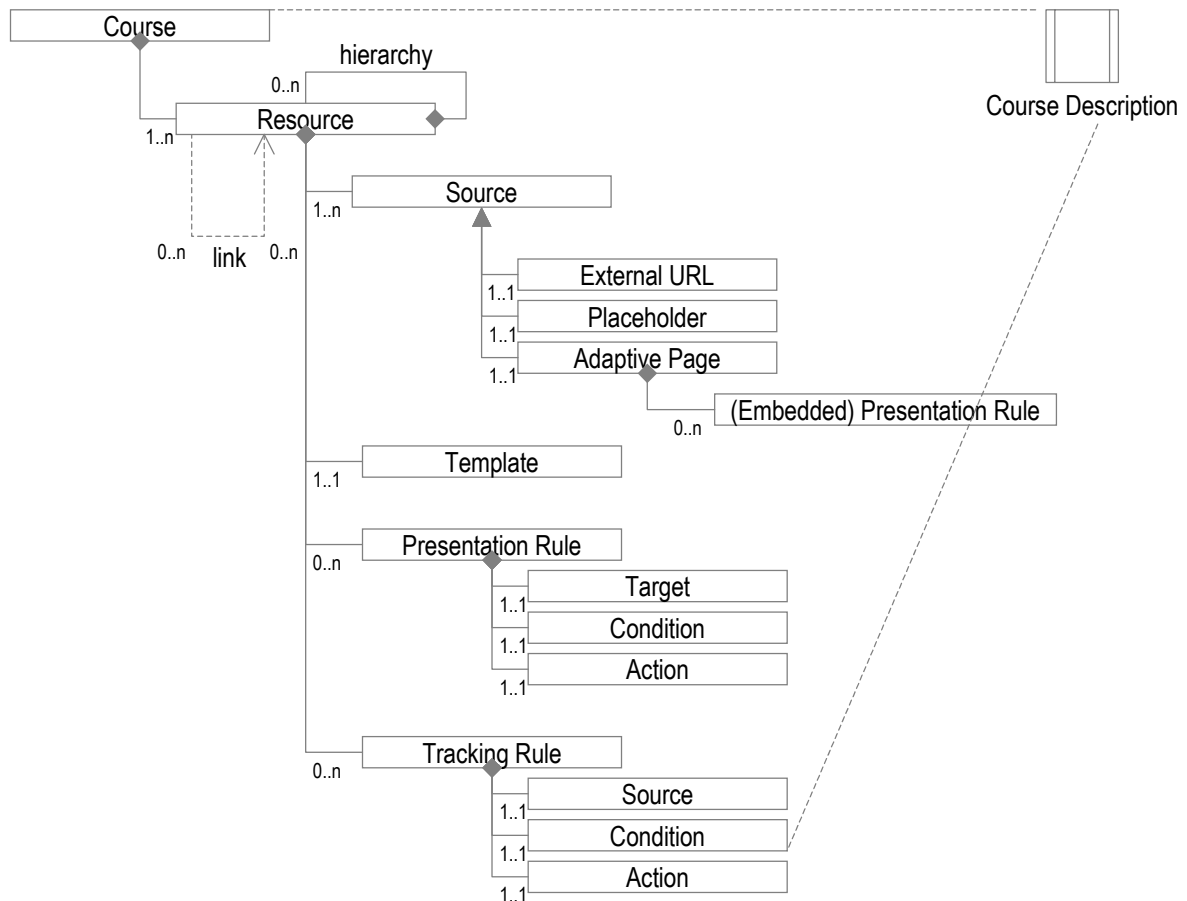


Figure 1. Data Model of ADLEGO

The Presentation Rules

ADLEGO supports the following three types of presentation rules:

1. rules affecting resource accessibility (*access rules*)
2. rules affecting fragment visualization (*fragment rules*)
3. rules affecting link visualization (*link rules*)

Access Rules. These rules are checked when a resource is requested by a client. The available actions are:

1. Grant access
2. Deny access
3. Redirect to another resource

Fragment Rules. When a fragment is found in a page, the ADLEGO system must decide how to render it according to the associated rule. Rule actions for a fragment are:

1. Hide
2. Show

Link Rules. When a link to a resource must be placed in a page, it is up to the engine to decide how to render it according to the specified rule set. Each resource can have rules applying to incoming links (*linkfrom rules*), and outgoing links (*linkto rules*). The actions that can be performed on a link are:

1. Hide: remove the link from the page
2. Mask: remove the anchor (but the link it is still visible), thus disabling it
3. Style(style): apply a specific style to the link (ex. Color, font, and so forth...)
4. Add a note(text): append a pop-up note to the link
5. Add an icon(icon): append an icon to the link

The Tracking Rules

A tracking rule has the following format:

Source – Condition – Action

The *source* refers to the source event that triggers the rule. A source event can be one of the following:

1. *Access-granted*: it occurs when an user successfully accesses the resource
2. *Access-denied*: it occurs when an user tries to access a forbidden resource (due to inhibiting presentation rules)
3. *User-redirected*: it occurs when an user is redirected to another resource
4. *User-created*: it occurs when a new user is added to the system
5. *User-login*: it occurs when an existing user logs into the system
6. *User-logout*: it occurs when an existing user logs out from the system using the logout button

Then, a tracking rule contains information about the action that must be performed. The action is basically a message that must be sent to an external user model. The action contains information on the target server and the message to send.

Obviously the body of the message is dependent on the type of messages that a specific user model accepts. The current implementation of ADLEGO is able to communicate with a custom user model we realized for research purposes, namely the Simple Knowledge Base (SKB) user model. The actions supported by SKB are shown in Table 2.

Table 2. Supported actions for communicating with Simple Knowledge Base user model

Command	Description
set_int	set a value for an integer variable
set_bool	set a value for an Boolean variable
set_string	set a value for an string variable
set_perc	set a value for an percent variable
inc_int	increment (or decrement) an integer variable
inc_perc	increment (or decrement) a percent variable
concat_string	concatenate a string variable with another string
flip_bool	flip the state of Boolean variable

The Authoring Process for ADLEGO

The ADLEGO system in itself does not solve the primary goal of disclosing adaptivity to non-technical teachers. Still too complex tasks are necessary to build a course with the system. On the other hand the great flexibility of ADLEGO must be ruled by the teacher in order to shape an adaptive course. For this reason an authoring tool is required to enable teachers designing and authoring their courses without (or at least with minimal) technical assistance.

We tackled the issue of defining an authoring tool by first analyzing what is the authoring process we must support with such tool. From user testing of ADLEGO, without the support of an authoring tool, we found that the most relevant and hindering tasks are those involving the creation and manipulation of rules, structure and contents. With more reason this happens because these tasks in ADLEGO require manipulating XML files, and adding XML statements in the ADLEGO syntax, which is a very hard and time consuming task. This process indeed is at least tedious for a programmer, but it is even impracticable for a non-technical instructor. For this reason we identified a minimal list of tasks that the authoring tool should support (cf. Table 3).

The table 3 was obtained by decomposition of macro authoring tasks (task of level 0) into more specific tasks (task of level 1 and 2). It is worth discussing the two complementary tasks of rule manipulation (“Manipulating presentation [or tracking] rule”). These tasks include the necessary tools to show the active rules for each resource, and to explain their meaning to the teacher. Since ADLEGO supports an order-dependent rule execution paradigm, the interface must support also rule sorting. Moreover, for rule editing, the interface should help the teacher picking up the right action, and writing down the necessary conditions.

Content editing as well, is an important task because we realized that ADLEGO syntax for fragment insert/removal can be misleading. This mainly happens because of the many kinds of tags that a page is

composed of. All of these issues contribute in disorienting the user while writing the required syntax of a fragment.

Table 3. List of Tasks to be Supported by the Authoring Tool

Task (level 0)	Task (level 1)	Task (level 3)
Setting the title of a course		
Structuring resources in a hierarchy		
Setting the title of a resource		
Setting the content of a resource	Selecting the type of content	
	Adding the content in the specified format	
Setting the graphic template of a resource		
Manipulating the presentation rules of a resource	Browsing the presentation rules	
	Adding a new presentation rule	
	Sorting the existing rules	
	Editing a rule	
	Deleting a rule	
Manipulating the tracking rules of a resource	Browsing the tracking rules	
	Adding a new tracking rule	
	Sorting the existing rules	
	Editing a rule	
	Deleting a rule	
Editing the presentation rule content	Defining the required target	
	Selecting the proper action	
	Defining the required condition	
Editing the tracking rule content	Defining the firing event (source)	
	Selecting the proper action	
	Defining the required condition	
Managing users	Adding users	
	Editing users	
	Removing users	
Editing content to a resource	Adding fragments	Inserting html content
		Defining action
		Defining condition for action
	Removing fragments	
Testing the application at design time		
Modify the application at runtime		

In addition to the specified tasks, an incrementally authoring process must be envisaged and therefore supported. Indeed, from our experience with the MAID methodology often the course structure is sketched before actually authoring the contents. Unfortunately this approach is prone with errors because of the necessary cycles that must be done to complete and polish the course. The most frequent kind of mistake is forgetting to feed some pages with actual content, leaving blank pages or dummy texts. For this reason special measures must be considered to check consistency and inform the teacher about possible flaws of the inserted content.

VIDET: a Visual Design Environment for Teachers

The Authoring Tool for ADLEGO is a visual multi-window interface that presents in an integrated view the different facets of an adaptive website: the hypertext structure, the content, the adaptive interaction model, the user model.

The interface consists of three panels (cf. Figure 2):

- Design panel (left)
- Preview panel (top-right)
- Help panel (bottom-right)

Moreover, several pop-up windows can be opened during the manipulation of elements of the interface, as we will show further on.

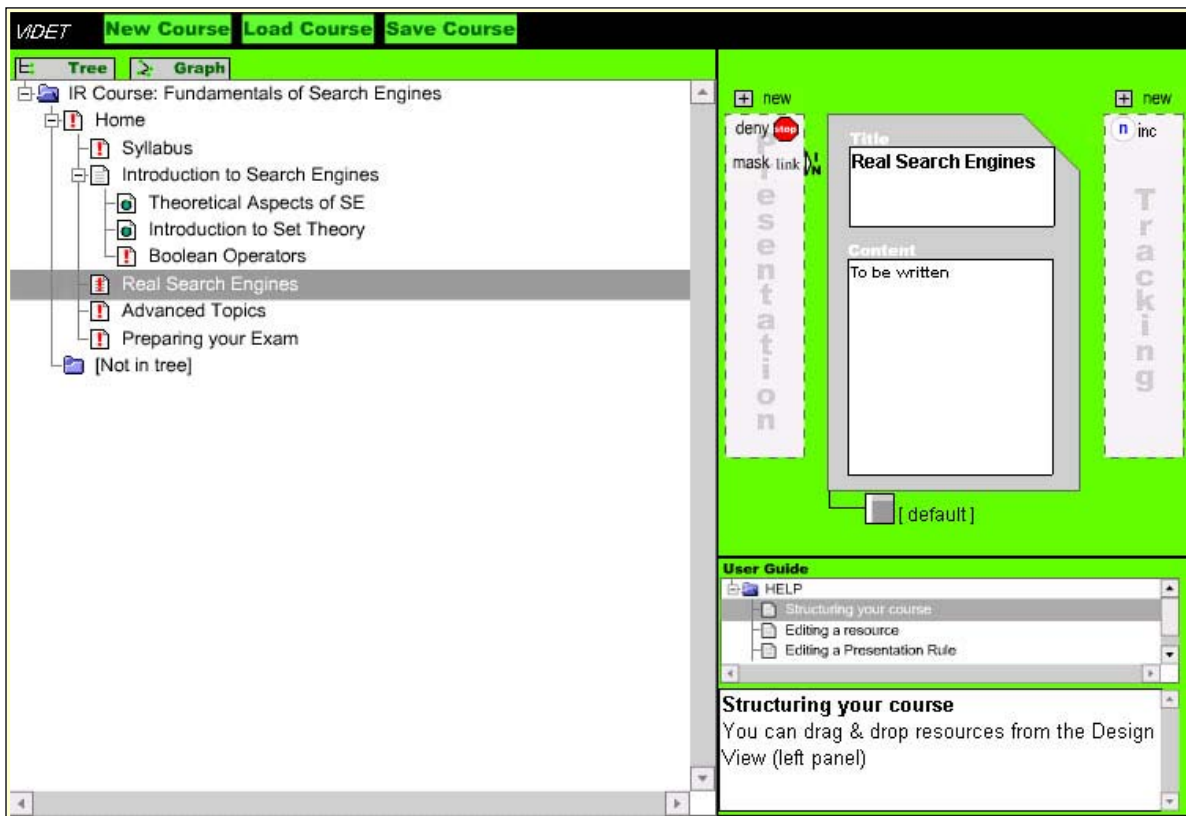


Figure 2. The VIDET Interface: the design panel on the left, the preview panel on the top-right, and the help panel on the bottom-right

The design panel gives an overview of the resources of the course and their structure. ADLEGO supports a traditional tree-like structure as many commercial Learning Management Systems nowadays do. Besides, it is possible to develop a more complex hypertext structure of resources by connecting them with hyperlinks. For this reason a complementary graph view of the course is provided. Within this interface the educator can add new resources and put them inside the tree with a simple drag & drop mechanism. Moreover some visual cues inform about the nature of the content of a resource.

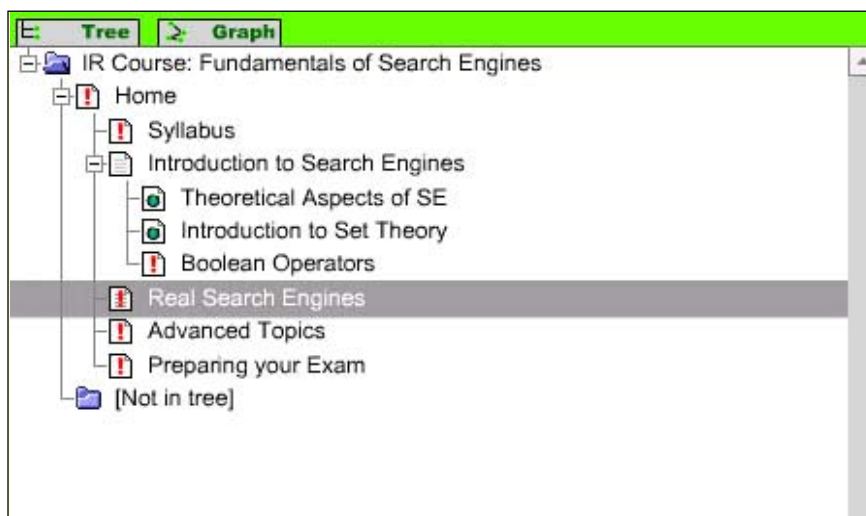


Figure 3. The Design panel of VIDET

The content types supported can be internal adaptive pages (the 'document' icon), external pages (the world icon), or placeholders (the 'exclamation mark' icon), as shown in Figure 3.

Placeholders are useful dummy texts than can be filled with instructions or reminders for authoring content at a later stage of the process. This feature is very useful for supporting fast prototyping activities. The exclamation mark icon provides quick information about which resources still need actual content.

Clicking on a resource in the left panel opens the corresponding resource detail in the preview panel (cf. Figure 4). In this panel a preview of the current resource is presented, displaying its title, content, and graphic template. Each of these elements can be directly modified in the preview, allowing in this way a quick and comfortable editing of contents.

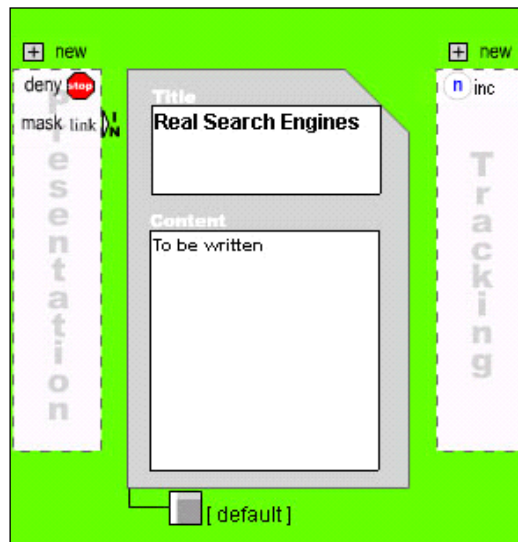


Figure 4. The Preview panel of VIDET

Each resource page can be rendered using a different template. The template can be chosen from a list of available templates, by the mean of the template preview window (cf. Figure 5).

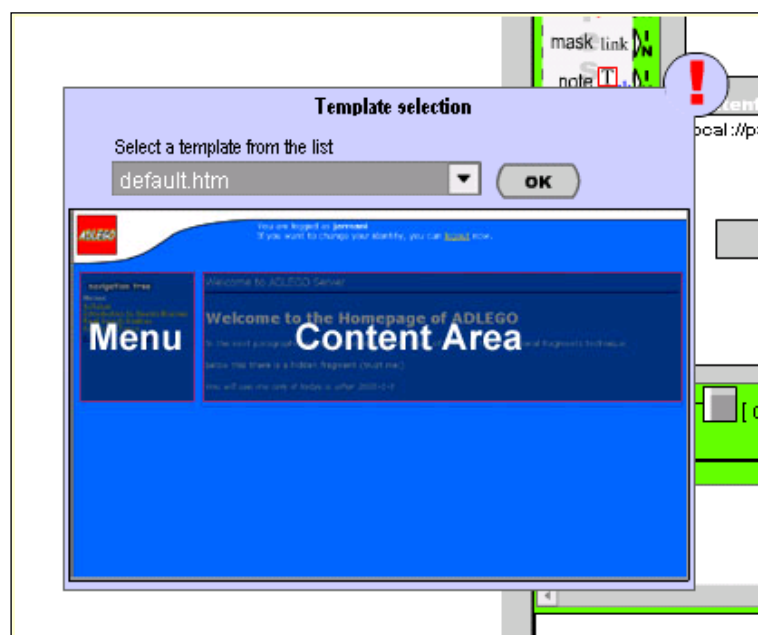















Figure 5. Template preview window

Along with the resource preview, all of the rules that have been specified are listed on the left and right side of it. As anticipated, the rules can be either presentation or tracking rules, which are respectively plotted in the left and right column. Each rule is displayed with an icon and a label to easily inspect the action that is performed when the rule condition applies. For link rules an additional cue informs whether the rule applies to incoming or outgoing links. Clicking a rule icon brings up a contextual menu to sort, edit, or delete the corresponding rule.

Table 4. Complete list of icons for presentation and tracking rules

Presentation rules		Tracking rules	
deny 	Deny access	 set	Set a Boolean value
hide 	Hide hyperlink	 flip	Flip a Boolean value
mask 	Mask (disable) hyperlink	 set	Set a string value
note 	Add a note to the link	 set	Set an integer value
icon 	Add an icon to the link	 inc	Increment an integer value
style 	Set a style for the link	 set	Set a percent value
			Increment a percent value

To edit a rule a Rule Editor is provided on a pop-up window (cf. Figure 7). The Rule Editor comes in two versions: basic, and advanced. The basic interface is intended to help teachers writing simple rules, by visually manipulating Boolean operators and variables, this way reducing the likelihood of errors during the formulation of conditions, and achieving at the same time better interpretation results than with the advanced interface. This approach stems from past studies on the difficulties encountered by non-programmers in the statement of Boolean conditions (Pane et al, 2001; Pane and Myers, 2000), and it aims to cope with the reported common misunderstanding of *not* and *or* Boolean clauses by non technical people.



Figure 6. Basic version of the condition editor

A Boolean condition is presented by explicitly parenthesizing its components. In this way we take care of the frequent user's misunderstanding of Boolean operator scopes.

The interface also ensures a visual type checking by showing the output type of each variable and operator, be Boolean, arithmetic or textual. Moreover variables are presented by a discursive textual label that explicitly conveys their semantics. All of these elements together contribute to support the interpretation task. In addition,

the formulation task is supported by providing a set of palettes that group all the available operators and variables which can be used to define a condition. The author may drag and drop any element from the palettes into the stage, this way incrementally defining a condition. The system automatically provides visual feedback, by highlighting the spots where an item can be dropped, therefore ensuring the type consistency. Therefore the underlying ADLEGO syntax is completely hidden to the user.

Conversely, the advanced interface supports the fully expressive power of the ADLEGO rule engine, yet providing a less error-prone environment for writing complex rules than using a common text editor (cf. Figure 7). This result is accomplished by a set of toolbars, which allow to quickly build a condition without knowing exactly the underlying syntax of each term. The instructor simply clicks on a variable icon, or a Boolean operator and the interface fills the condition window with the corresponding statement according to the right syntax. Finally, the expert user can also type a condition using the ADLEGO syntax and test it with the syntax checker.

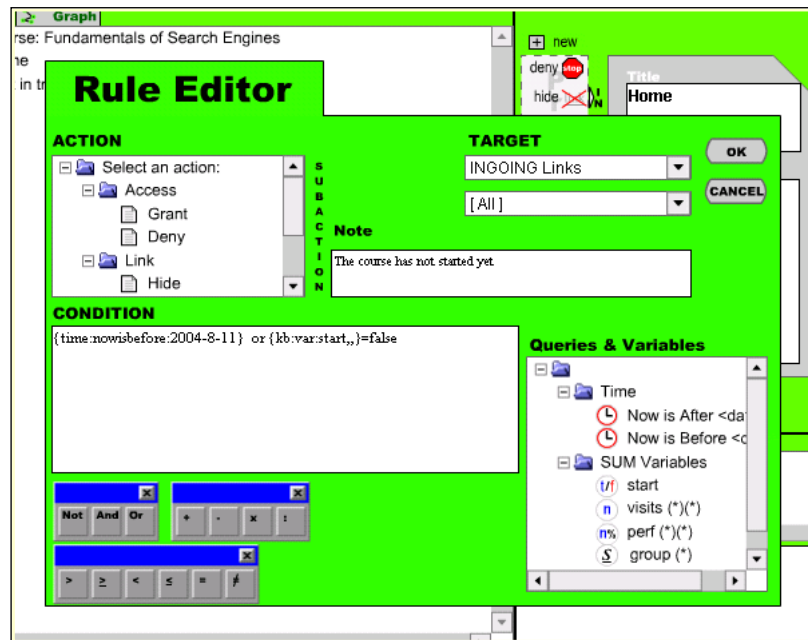


Figure 7. Advanced version of the condition editor

A similar interface is provided for writing tracking rules that however use a different set of actions: setting, incrementing, decrementing the value of a variable, etc.

VIDET features also an easy to use fragment editor to edit adaptive pages based on techniques of fragment inclusion/removal. The fragment editor supports basic word-like operations for formatting and aligning text, adding images and hyperlink. Besides, it supports the fragmentation of content. Each fragment can be targeted by a rule, stating its show (or hide) conditions. To ensure the interface consistency, fragment rules share the same condition editor of the presentation and tracking rules (cf. Figure 7).

VIDET fragment editor cannot compete with html editors such as Macromedia Dreamweaver, or Homesite, but it is designed to provide plain and sound editing features anyway. Moreover it allows importing existing html pages produced with external packages, and slicing them into fragments that then can be hidden or shown.

Finally VIDET supports designers with an online manual that explains the necessary steps to perform the most frequent tasks with the tool (cf. Figure 9). The manual is organized by tasks in a hierarchical structure, which is expandable and collapsible by the user. For each task the manual gives a quick description of relevant commands to perform it. The current implementation supports only a static online manual, but we foresee to design an intelligent version, that automatically picks up the relevant information for the current designer's task.

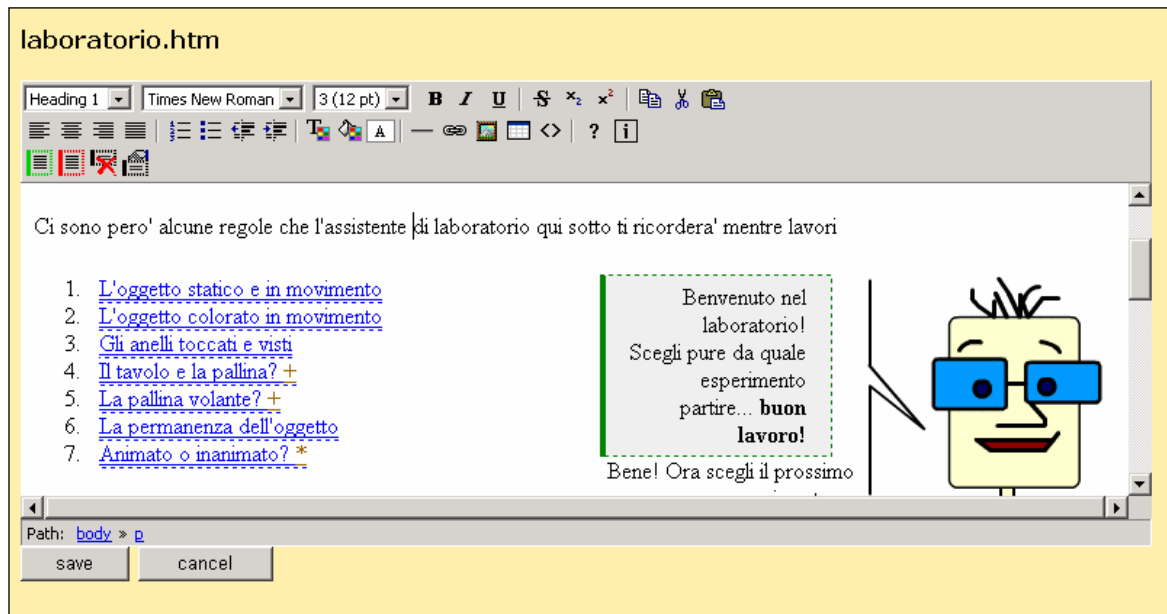


Figure 8. Fragment editor of VIDET

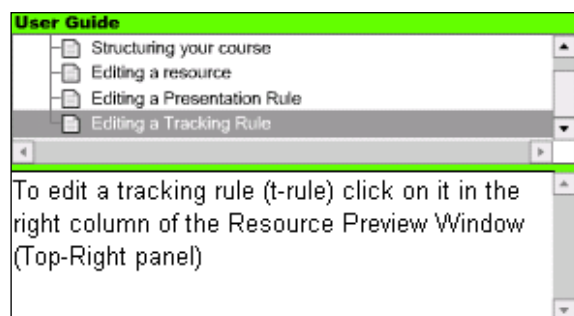


Figure 9. The Help panel of VIDET

Implementation of the Tool

VIDET has been implemented as a prototype for usability testing. The current implementation exploits Macromedia Flash 6 as the main development environment. The choice of Flash instead of more traditional development environments (such as Microsoft .Net, Sun JDK, etc.) derives from the evaluation aims we had. Our primary objective was designing and quickly evaluating different visual solutions to present complex concepts and operations to non-technical designers. Since this aim must be tackled by a trial & error approach (unfortunately there are no ready-to-use recipe on the topic!), we needed a development environment with strong visual features, and a huge library of components and behaviors. Flash responds perfectly to these requirements.

The Flash implementation adopts the HTTP protocol to communicate with the ADLEGO server in order to load and save information about the course being created. Moreover it allows creating html pages and loading them directly into the ADLEGO directory without needing external FTP software.

Even though Flash 6 supports a raw object-oriented paradigm, still we were able to implement an OO-driven solution for the authoring tool. Basically VIDET uses three classes (i.e. course, resource, rule) to deal with all of the aspects of the ADLEGO data model. A XML parser has been implemented to read course information from ADLEGO, and create the necessary objects in VIDET. Then the visual interface allows the designer to create new instances of each object (namely, new resources, and new rules) and edit their properties. A XML exporter class is then in charge of translating the resource and rule instances into a XML description file that is uploaded to ADLEGO. This OO approach mirrors the underlying ADLEGO implementation, greatly simplifying synchronization of features from the adaptive engine to its authoring tool. Every new feature of ADLEGO is easily mapped into VIDET just by adding the necessary functions to the XML parser and exporter, and extending the corresponding classes of objects (resources, and rules).

So far, an unresolved task is finding a technical solution to deal with the issue of interacting with different user models other than the one that is built-in in ADLEGO. Since ADLEGO supports virtually any external user model, we should envisage that its authoring tool does the same. The main issue has a two-fold nature: a first problem is how to find a visual approach that works for every user model, without needing to hard-code a custom solution for each user model. Secondly, it is important to define a protocol to exchange information with such user models. The current implementation of VIDET just solved the latter, by adopting a XML-based protocol to exchange information with virtually any user model, which is able to understand XML as a communication protocol.

Evaluation

The evaluation of VIDET was a critical part of this work, since we wished to validate our research hypothesis to draw some conclusions. On the other hand, the effectiveness of a tool is just one facet of all the usability dimensions that characterize the user's experience with the tool. Therefore specific techniques must be planned to test each of these dimensions. For this reason we chose to adopt the MiLE methodology (Triacca *et al.* 2004) to evaluate VIDET. In fact, MiLE encompasses a sufficiently broad spectrum of usability techniques to achieve very different usability evaluation goals, ranging from measures of effectiveness to technical issue detection. Moreover, a nice quality of MiLE is its scalability and adaptability to different situations: indeed its core methodology can be adapted to fit any constraint of time, resources, and context.

MiLE is based on the combination of two complementary activities:

- **Expert review** (also known as *usability inspection*): this activity is performed by an usability expert who is in charge of thoroughly exploring the application.
- **User test**: real users are in charge of performing tasks on the tool, in order to detect possible flaws from the user's point of view.

For the inspection the evaluator used usability checklists and heuristics to highlight the most common technical issues, and to detect possible flaws in the user experience. The results of this activity led to the formulation of possible usability issues.

The subsequent user testing phase was organized both to assess the identified issues with actual users of the application, and to discover new undetected flaws. Since we expected that the user-VIDET interaction would be greatly affected by the user's background on Information Technology (IT), we decided to split the participants in two groups according to their skills in IT. Then we set up the same set of goals, and tasks for each of them, in order to compare their responses to VIDET. To meet project deadlines we limited the number of users per group to one, finding very interesting result, yet evidently not easy to generalize.

The results from both of the phases were recapped into a set of user experience indicators (UEIs), which may help designers making decisions for improvement.

To our concern, effectiveness is the most relevant indicator. Both quantitative and qualitative measures of effectiveness show that the tool is effective at supporting the author to produce an adaptive course. In particular, VIDET met the expert user's expectations, who find it flexible and efficient enough to create an adaptive course for ADLEGO. The basic user showed a more cautious attitude, because of the numerous and structured tasks that must be performed to define the interaction model of an adaptive course, and above all because of the programming nature of these tasks. Yet the basic user was able to perform all of the given tasks, after adequate training, suggesting that learnability of the tool is possible. In particular, a specific training must be envisaged to support users in the creation of the necessary items that enable an adaptive interaction (i.e. presentation rules, tracking rules, conditions, fragments). A useful way to achieve this could be to provide some step-by-step tutorials that show the creation of a sample of adaptive course to the new users.

Other aspects require some minor changes, for instance the tracking rule editor suggests a slightly different interaction flow than what the user may expect.

Discussion

According to results from HCI field (Hackos, 1999), four minimalism principles should be taken into account for the design of a user interface:

1. Choose an action-oriented approach;
2. Anchor the tool in the task domain
3. Support error recognition and recovery
4. Support ready to do, study, and locate

Taking them in account means basically asking ourselves a question: what should be the border between flexibility and usability of the interface? There is no an easy answer to that. In our project, we have narrowed down the spectrum of the targeted end users of our tool to be able to come to a conclusion. Our research framework is that the design process of an adaptive educational website is under the responsibility of an instructional designer with no programming background. This assumption leads to reduce as much as possible the richness of the interface, and therefore of the design blueprint that can be produced with it. On the other hand, it is evident that we cannot impoverish over a certain threshold the set of tools which an educator should be able to use. This issue still requires more debate, and we suggest that interesting results could be gained from the empirical usability evaluation of both adaptive courses and their authoring tools as well.

Another insight we had is that making use of strong visual metaphors for helping the educator understanding all the aspects of an adaptive system, would improve the usability of the authoring tool in the end. Yet, what is the best metaphor to achieve this? The current project makes use of an enhanced “desktop” metaphor to build up on the existing conventions that we all share by the nature of modern window based operative systems. Other metaphors could suit the case as well.

Moreover, since the design of an user-system interaction requires a time consuming task for testing that all of the designed behaviors actually occur, some kind of simulation or debugging tools should be provided to asses it in a controlled and organized way.

The current implementation of the authoring tool has been designed to suit the needs of the specific adaptive engine, namely ADLEGO, however a more general approach could be explored as well. The next generation of authoring tools could be able to produce a more application-independent model, and could provide compiling tools to translate it into the major adaptive engines (i.e. the same authoring interface could compile the same model of adaptive course in either the Interbook, AHA, or ADLEGO syntax).

Finally a more complex scenario arises when we think in the perspective of web services. Imagine a network of: adaptive servers (executing a different adaptive engine each), user models, and content and service providers. In this context the authoring and design needs cannot be easily integrated in a holistic interface anymore. But still, the designer of the learner experience (our educator), needs to have access to all of them, and perhaps even modify parts of them.

Related Works

VIDET is straightforward, but powerful authoring tool tailored to non technical people. Other proposals in fact misses the point of tailoring the tool to their real users, namely educators.

For instance, AHA! (De Bra *et al.* 2003) comes in bundle with a very limited set of authoring tools, which are too abstract and textual to allow the required manipulation a non-programmer would need. The ALE environment (Kravcik *et al.* 2004) suffers from the same issue, giving authors only form-based and textual authoring tools. More complex tool such as Eon (Murray 2003), although they visually assist an author to tweak all of the aspects of an ITS, they are not specifically tailored to non-technical users.

Conclusion and Future Work

A visual authoring tool to design an adaptive website seems to be the missing key to unfold the hidden potentiality of adaptive technologies for education. The importance of visualizing all of the different aspects of educational adaptive systems is even more evident when we think at the community of instructional designers and teachers as our main target audience. Those members do not usually have the required technical skills to unleash the power from an adaptive engine, even though it has been designed with simple features (i.e. conditional fragments, adaptive link annotation only).

A prototype of a visual authoring tool for instructors has been discussed in this paper. The authoring tool allows a teacher to structure the website in a network or a tree of resources, connect them with the actual content (both locally stored or coming from an external website), and define the necessary rules to present it in an adaptive form to the learners, just using simple drag & drop mechanisms. Moreover every part of the adaptive course being developed is shown to the author in a visual form, in order to support them thinking and manipulating the usually abstract objects that constitute an adaptive website (concept networks, user model variables, etc.). An evaluation featuring usability inspections and user tests was performed. Results showed that VIDET succeed at supporting non-technical authors of adaptive websites, yet some training is unavoidable.

Since the authoring tool is still under development, some features have not been implemented yet. In the near future, the tool will allow to store and retrieve sub-sets of resources, along with their rule sets, to foster reusability. This feature will pave the road to the development of design patterns for adaptivity.

Acknowledgments

The current work was carried on under the supervision of Prof. Brusilovsky (School of Information Sciences, University of Pittsburgh, PA, USA).

References

- Ainsworth, S. E., Major, N., Grimshaw, S. K., Hayes, M., Underwood, J. D., Williams, B., & Wood, D. J. (2003). REDEEM: Simple Intelligent Tutoring Systems From Usable Tools. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environment*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 205-232.
- Armani, J., & Botturi, L. (2003). Adaptive Hypermedia System Design: a Method from Practice. *Paper presented at the IADIS International Conference on WWW/Internet 2003*, November 5-8, 2003, Algarve, Portugal.
- Armani, J. (2004). Shaping Learning Adaptive Technologies for Teachers: a Proposal for an Adaptive Learning Management System. *Paper presented at the IEEE International Conference on Advanced Learning Technologies*, August 30 - September 1, 2004, Joensuu, Finland.
- Aroyo L., Dicheva D., & Cristea A. (2002). Ontological Support for Web Courseware Authoring. *Paper presented at the International Conference on Artificial Intelligence (ICAI'02)*, June 24-27, 2002, Las Vegas, USA.
- Arruarte, A., Ferrero, B., Fernández-Castro, I., Urretavizcaya, M., Álvarez, A., & Greer, J. (2003). The IRIS Authoring Tool. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environment*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 233-268.
- Avgeriou, P., Vogiatzis, D., Tzanavari, A., Retalis, S. (2004). Design Patterns In Adaptive Web-Based Educational Systems: An Overview. *Paper presented at the Web Education Conference 2004*, February, 11-14, 2004, Innsbruck, Austria.
- Botturi, L. (2003). E²ML - A Modeling Language for Technology-dependent Educational Environments. *Paper presented at the World Conference on Educational Multimedia and Hypermedia*, June 22-28, 2003, Honolulu, Hawaii, USA.
- Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User Adapted Interaction*, 11 (1/2), 87-110.
- Brusilovsky, P. (2003). Developing Adaptive Educational Hypermedia Systems: from Design Models to Authoring Tools. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environment*, Dordrecht: Kluwer Academic Publishers, 377-410.
- Brusilovsky, P., Eklund, J., & Schwarz, E. (1998). Web-based education for all: A tool for developing adaptive courseware. *Computer Networks and ISDN Systems*, 30 (1-7), 291-300.

- Carro, R. M., Pulido, E., & Rodrigues, P. (2000). How Adaptivity Affects the Development of TANGOW Web-Based Courses. In P. Brusilovsky, O. Stock, & C. Strapparava (Eds.), *Lecture Notes in Computer Science, 1892*, Berlin Heidelberg: Springer-Verlag, 280-283.
- Ceri, S., Fraternali, P., & Bongio, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks, 33* (1-6), 137-157.
- Coda, F., Ghezzi, C., Vigna, G., & Garzotto, F. (1998). Towards a Software Engineering Approach to Web Site Development. *Paper presented at the 9th IEEE International Workshop on Software Specification and Design (IWSSD)*, April 16-18, Ise-shima, Japan.
- Cristea, A. I., & De Mooij, A. (2003). LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. *Paper presented at the Twelfth International World Wide Web Conference*, May 20-24, 2003, Budapest, Hungary, retrieved July 15, 2005, from <http://www2003.org/cdrom/papers/alternate/P301/p301-cristea.pdf>.
- De Bra, P., & Calvi, L. (1998). AHA! An Open Adaptive Hypermedia Architecture. *The New Review of Hypermedia and Multimedia, 4*, 115-139.
- De Bra, P., Houben, G. J., Wu, H. (1999). AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. *Proceedings of the ACM Conference on Hypertext and Hypermedia*, Darmstadt, Germany, 147-156.
- De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., & Stash, N. (2003). AHA! The Adaptive Hypermedia Architecture. *Proceedings of the 14th ACM conference on Hypertext and hypermedia*, New York, NY, USA: ACM Press, 81-84.
- EduTools (2004). *Providing decision-making tools for the E-D-U community*, retrieved July 15, 2005, from <http://www.edutools.info/course/>.
- Finzer, W. F., & Gould, L. (1993). Rehearsal World: Programming by Rehearsal. In Cypher A. (Ed.), *Watch What I Do - Programming by Demonstration*, Cambridge, England: The MIT Press, 79-100.
- 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.
- Hackos, J. T. (1999). An Application of the Principles of Minimalism to the Design of Human-Computer Interfaces. *Common Ground, 9*, 17-22.
- Isakowitz, T., Stohr, E. A., & Balasubramanian, P. (1995). RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM, 38* (8):34-43.
- Kravicik, M., Specht, M., & Oppermann, R. (2004). Evaluation of WINDS Authoring Environment. In P. De Bra, & W. Nejdl (Eds.), *Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems: Third International Conference (AH 2004)*, Berlin: Springer-Verlag, 166-175.
- Kupka, T., Zajaczek, J. E. W., Behrends, M., Walter, G. F., & Matthies, H. K. (2004). Schoolbook - An Authoring Tool and Content Management System. In V. Uskov (Ed.), *Proceedings of Web Education Conference 2004*, Calgary, AB, Canada: ACTA Press, 169-171.
- Murray, T. (2003). Eon: Authoring Tools for Content, Instructional Strategy, Student Model and Interface Design. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environment*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 309-339.
- Pane J. F., & Myers, B. A. (2000). Tabular and Textual Methods for Selecting Objects from a Group. *Proceedings of the IEEE International Symposium on Visual Languages*, Seattle, WA: IEEE Computer Society, 157-164.
- Pane J. F., Ratanamahatana C. A., & Myers B. A. (2001). Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems. *International Journal of Human-Computer Studies, 54* (2), 237-264.

- Park, O., & Lee, J. (2004). Adaptive Instructional Systems. In Jonassen, D. H. (Ed.), *Handbook of Research on Educational Communications and Technology* (2nd ed.), Mahwah, NJ: Lawrence Erlbaum Associates, 651-684.
- Schwabe, D., & Rossi, G. (1998). An Object Oriented Approach to Web-Based Application Design. *Theory and Practice of Object Systems*, 4 (4), 207-225.
- SCORM (2004). *Best Practices Guide for Content Developers*, retrieved July 15, 2005, from <http://www.lsal.cmu.edu/lsal/expertise/projects/developersguide/>.
- Sparks, R., Dooley, S., Meiskey, L., & Blumenthal, R. (2003). The Leap Authoring Tool: Supporting complex courseware authoring through reuse, rapid prototyping, and interactive visualizations. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environment*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 411-438.
- Stash, N., Cristea, A., & De Bra, P. (2004). Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions. *Paper presented at the WWW 2004 Conference*, May 17–22, 2004, New York, NY, USA.
- Triacca, L., Bolchini, D., Botturi, L., & Inversini, A., (2004). MiLE: Systematic Usability Evaluation for E-learning Web Applications. *Paper presented at the EDMEDIA 2004 Conference*, June 21-26, 2004, Lugano, Switzerland.
- Van Joolingen, W. R., & de Jong, T. (2003). SimQuest: Authoring educational simulations. In T. Murray, S. Blessing, & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environment*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1-31.
- Weber, G., Kuhl, H.-C., & Weibelzahl, S. (2001). Developing adaptive internet based courses with the authoring system NetCoach. *Paper presented at the Third workshop on Adaptive Hypertext and Hypermedia*, July 14, 2001, Sonthofen, Germany.