

Interactive Design of Adaptive Courses

J. A. Macías and P. Castells

E.T.S. de Informática, Universidad Autónoma de Madrid

Keywords: Authoring Tools, Interactive Design, Adaptive Software, Task Modeling, User Modeling, Distance Learning, Human-Computer Interaction

Abstract: The ability of educational applications to adapt to individual students has been a sought-after feature for more than a decade. The difficulty of developing this kind of software and the lack of adequate tools that facilitate this endeavor have hindered the participation of instructors and teachers in the elaboration of adaptive applications. In this paper we describe an authoring tool, ATLAS, that aims at combining expressive power and ease of use in the design of adaptive web courses. ATLAS allows the fully interactive construction of courses that adapt automatically to the student's characteristics and her/his behavior while taking the course. The designer interacts with the tool by using an intuitive visual language based on the direct manipulation of elements involved in the course. The tool takes care of the transition between a teacher's understanding of the course and the representation model of the underlying system.

1. INTRODUCTION

With the rapid introduction of the WWW technology in the educational field, educational applications are facing an increasingly wide and heterogeneous audience. In this context, the diversity of needs and particularities in such large communities of learners is becoming a first-order concern. Since it is impossible to satisfy the preferences and needs of all students with a single solution, attempts are being made to produce flexible software, able to adapt to different kinds of users, platforms, and situation. Given its complexity, this kind of software is extremely difficult to develop,

which hinders the direct, active involvement of education professionals in the elaboration of the product, to the detriment of its pedagogical quality.

A wide variety of interactive authoring tools and easy-to-use graphical environments are available today that support diverse aspects of courseware authoring, like WebCT [1], Macromedia Director, or HTML editors, to name a few. These tools greatly simplify the development of educational materials, but they do not provide an adequate support for the dynamic aspects of an adaptive course. Moreover, the type of assistance they provide tends to be confined to work over partial aspects of a course, or a subset of aspects, in isolation, like multimedia components, simulations, static tables of contents, or web-site administration, in such a way that the dynamic integration of all these elements is done using supplementary non-interactive tools or programming languages.

On the other hand more powerful tools of a different kind have been developed during the last few years that do support the construction of interactive courses with a high degree of flexibility, including adaptive capabilities, using a relatively small set of high-level primitives [2, 3, 4]. However these tools are not easy to use in general, as they require learning special-purpose specification languages, and understanding non-trivial abstract concepts that are out of reach for authors who are not necessarily familiar with the complexities of software development. Complementing this type of tools with editors based on intuitive visual languages, to isolate the designer from the complexity of the underlying system, is a difficult problem when building dynamic courses, since the course as such does not exist at design time.

In this paper we describe a design tool, ATLAS (Authoring TooL for Adaptive Software design), for the interactive construction, in a graphical editing environment, of interactive courses with the capability to adapt dynamically to students. ATLAS helps courseware authors get a better understanding of the different aspects of an adaptive course and the relations among them: structure, contents, presentation, and student model, by integrating the design of all these elements in a single visual environment. To reduce the need for abstraction from the author, the tool offers the option of working at design time on a representation of courses that is close to the final result, by using concrete examples of the student model, simulated by the designer.

Our system aims at bringing the advanced techniques for adaptive course construction recently developed in the field of computer-assisted learning within reach of instructors and subject-matter experts that do not have a strong computer science background. ATLAS combines the expressive power of a system based on the abstract modeling of the student and his/her learning tasks, with the ease of use of graphical specification based on direct

manipulation, relieving the designer from using textual specification languages.

ATLAS is being developed on top of a preexisting tool, TANGOW [4], that is used as a support system for modeling and executing the adaptive courses defined by the designer in the ATLAS environment (see figure 1). The current version uses the WWW as the platform for the execution of the courses.

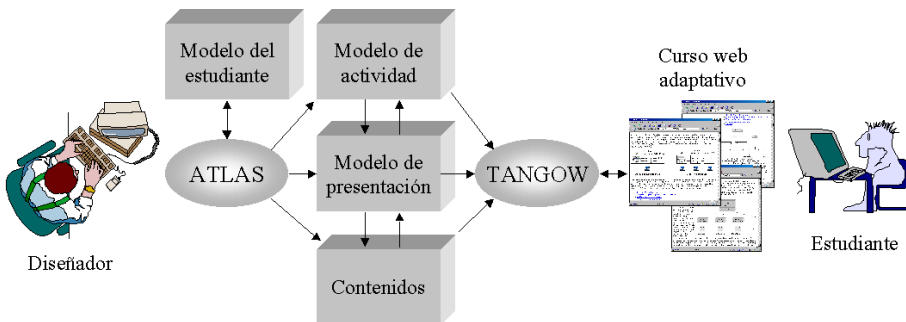


Figure 1. System architecture

2. RELATED WORK

Adaptive hypermedia support technology has made a considerable progress during the last decade, in particular in the field of Computer-Assisted Learning, with systems like DCG [2], ELM-ART [3] and TANGOW [4]. ELM-ART and DCG use an explicit representation of domain concepts, interrelated in a prerequisite graph. DCG includes a planner that guides the student along a path to reach goal concepts starting from already known concepts. ELM-ART uses a sophisticated system to estimate the knowledge acquired by the user in relation to a concept map of the course, according to which the system dynamically proposes the student a path to follow at each moment. While in ELM-ART and DGC the structure of courses is fixed, being the student itinerary what varies, TANGOW generates the course structure at runtime. TANGOW models student activity in the form of a hierarchy of tasks that represent didactic units the student can perform. Contents are associated to tasks by assigning a set of HTML fragments to each task, from which web pages are generated when the student is ready to undertake a certain task. Hierarchy among tasks establishes that the accomplishment of certain tasks consists of carrying out other subtasks, which in turn may be composite or not. The structure of the

task tree is dynamic and depends on student characteristics and her/his behavior while interacting with the course.

From the point of view of the designer, adaptive course modeling in TANGOW can be compared to the description of user tasks in model-based user interface construction tools like UIDE [5], MASTERMIND [6] and MOBI-D [7]. Aimed at a wider scope beyond the educational domain, these systems support the construction of user interfaces from a modelization of the tasks that the user must be able to carry out with the application being built. Both UIDE and MOBI-D include facilities for the graphical manipulation of user tasks. MASTERMIND does not provide a visual editing environment for task models, but provides a very powerful specification language for tasks. The main novelty in our work with respect to these systems is the explicit manipulation of a user model in the development environment itself.

Relating user models to task models requires adequate abstraction capabilities from the specification language. Dealing with abstraction in a graphical environment raises new difficulties. Visual languages are appropriate to transmit declarative information, but it is not easy to describe dynamic behavior this way. Programming by Example systems [8] solve this problem by inferring procedural information from concrete examples created by the designer. We have carried this idea to our context, in particular the techniques used in HANDSON [9], a tool that allows the visual construction of dynamic interface presentations that depend on any parameter of the application, the platform of the interface itself.

3. THE ATLAS AUTHORIZING TOOL

ATLAS takes care of the transition between the teacher's view of the course and the representation model of the underlying system. In this sense ATLAS can be considered as a user interface for the course designer in TANGOW. The development environment presents all the elements of the course to the designer (tasks, contents, student model) as objects on the screen that the designer can create, manipulate, and associate to each other directly with the mouse. Figure 2 shows an intermediate state in the creation, in this environment, of a course about Java programming. The workspace is formed of three main areas: task model, contents model and student model. In the first one (left side), the designer defines the structure of the course by building a model of student tasks which, in accordance with the TANGOW model, are organized hierarchically. The contents area (upper right) shows a catalog of HTML fragments that store the contents of the different parts of the course. In the student model area (lower right), student characteristics

that are relevant for the course being constructed are described. This model consists of a set of simple and composite attributes, forming a tree, that the designer can edit and use for the definition of conditions that will determine the course structure.

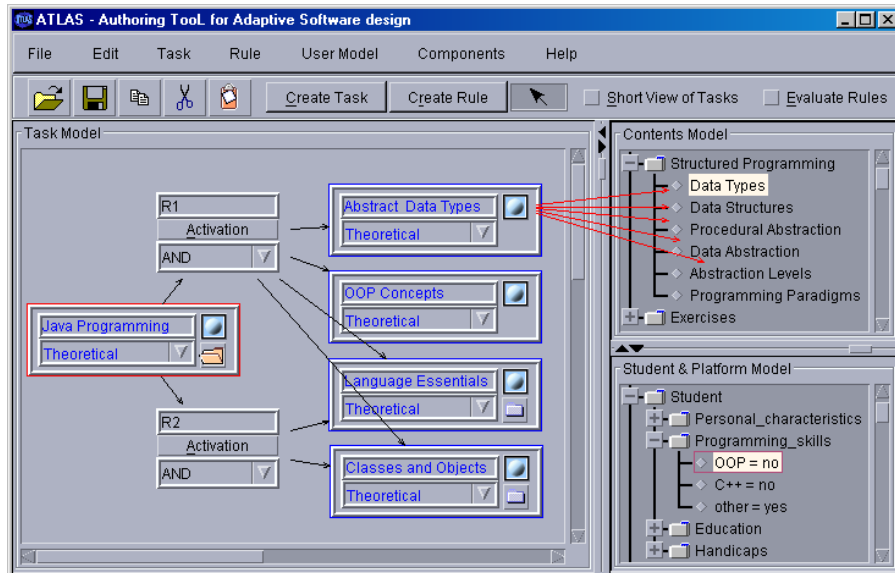


Figure 2. The ATLAS design environment

The designer assigns each task the content units that correspond to it, by dragging objects with the mouse from the course contents area onto the corresponding task. The task tree is constructed by dragging the mouse between tasks, associating each task with its subtasks. In figure 2, the designer has decomposed a high-level unit (task) about *Java Programming* into four subtasks: *Abstract Data Types*, *OOP Concepts*, *Language Essentials*, and *Classes and Objects in Java*.

Tasks admit more than one subtask decomposition. Between each task and its subtasks a node is interposed (*R1* and *R2* in the figure) for each alternative ramification. These objects represent the adaptive element by which the structure of the course is made dependent on students' characteristics and their activity during runtime (see [4]). Their role consists of controlling which decomposition will apply at execution time when, like in the example, more than one alternative has been set. The activation of one or the other is determined by a predicate that is associated to each branching-node. These conditions are edited by selecting and dragging attributes from the user model onto a dialog box where conditions are edited. This dialog box is opened by double-clicking on the icon that represents the conditional

branching-node (see figure 3). ATLAS allows combining simple comparison predicates between student properties, task parameters and/or literal values, in normal disjunctive form.

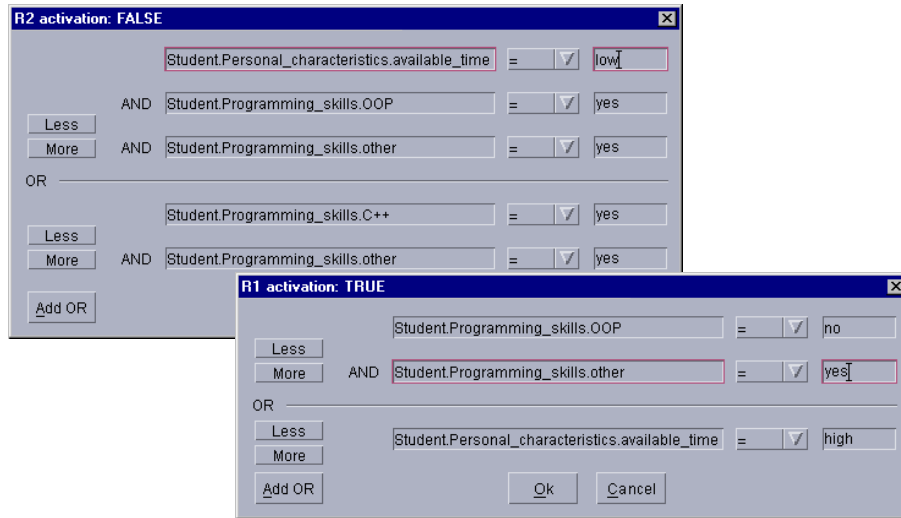


Figure 3. Constructing activation conditions

To make it simpler for the designer to deal with adaptive aspects, ATLAS allows working on specific user profiles provided by the designer, so that the tool infers generalized task models from partial specifications built for particular cases. To do this, the designer first edits the student model, from which the structure of tasks depends, adding new properties or removing them, and introducing values. From the examples provided this way, ATLAS instantiates the task hierarchy for the described profile, allowing the designer to edit directly the resulting structure, instead of the original representation, more abstract and hard to understand. In figure 2 this option is turned off. When “Evaluate Rules” mode is turned on in the toolbar, branching *R2* disappears for task *Java Programming*, because its activation condition is false (figure 3) for the provided student model attribute values. When the designer works in this mode, the system takes care of generalizing the design and maintaining consistency.

ATLAS uses visual feedback to show the relations between the different model elements. Task to subtask, task to contents, and branching-node to student attribute connections appear as arcs between objects on the screen. In a realistic course with hundreds of interrelated tasks, this can become an extremely messy and unwieldy structure. To alleviate this problem and facilitate the navigation over large models under construction, ATLAS allows authors to select different levels of detail for different parts of the

model. For example, connection arcs can be shown or hidden on a per-task basis. The amount of information displayed in each object can vary (option “Short View of Tasks”). The designer can also collapse or expand entire task subtrees with a single click. For instance in figure 2, the folder icon located at the lower right corner of task *Java Programming* indicates that the task is expanded, while *Language Essentials* and *Classes and Objects* do not show their subtasks. The remaining two tasks are not composite, so they don’t have this icon.

The output from ATLAS is a model of the course ready to be processed by TANGOW, which provides runtime support. TANGOW generates web pages on the fly by executing branching conditions, assembling content fragments associated to tasks, and updating task state in response to student actions.

4. CONCLUSIONS

The development of high-quality educational products requires the direct involvement of professional educators in the elaboration of educational material. Beyond the selection or elaboration of text and multimedia resources, it is important for this involvement to have an effect on the definition of aspects like structure and functional characteristics of courses, as well as the interaction between the software and the student. We believe that the development of interactive tools that expressly support these aspects will contribute to the advance in this direction, allowing designers to spend more time and attention on pedagogical aspects, and less in solving technical problems.

The adaptivity of constructed applications allows for a wider reach for courses, favoring their diffusion over the net. Being aware of students’ characteristics and the way they work is a key element in education quality, be it computer-supported or not. The description and explicit use of user models in a graphical environment to define adaptive capabilities in the constructed applications is a novel aspect both in the educational technology field and in user interface development.

There are many possibilities for continuation of the presented work. For instance, course contents are currently elaborated using HTML editors that are external to the system. Our plans for the near future include the development of a tool for content authoring that supports adaptive presentation. We are also planning to incorporate into the system other models that should influence course structure and presentation, such as the platform, context of usage, or the characteristics of the data handled by the application.

ATLAS is being developed in Java, JDK 1.2. A version of the tool is currently available at <http://astreo.ii.uam.es/~atlas>.

5. ACKNOWLEDGEMENTS

The work reported in this paper is being partially supported by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL1999-0181.

6. REFERENCES

1. M.W. Goldberg and S. Salari. *An update on WebCT (World-Wide-Web Course Tools)- a Tool for the Creation of Sophisticated Web-based Learning Environment*. Proceedings NAUWeb'97 - Current Practices in Web-based Course Development. Flagstaff, Arizona, 1997.
2. J. Vassileva. *Dynamic Courseware Generation on the WWW*. Proceedings 8th World Conference of the AIED Society. Kobe, Japan, 1997, pp. 498-505.
3. G. Weber and M. Specht. *User modeling and Adaptive Navigation Support in WWW-based Tutoring Systems*. Proceedings 6th International Conference on User Modeling (UM97). Sardinia, Italy, 1997.
4. R.M. Carro, E. Pulido, P. Rodríguez. *Dynamic generation of adaptive Internet-based courses*. Journal of Network and Computer Applications, v. 22, 1999, pp. 249-257.
5. N. Sukaviriya, J. Foley, T. Griffith. *A Second Generation User Interface Design Environment: the Model and the Run-Time Architecture*. Proceedings ACM International Conference on Computer-Human Interaction (InterCHI'93). Amsterdam, 1993, pp. 375-382.
6. P. Szekely, P. Sukaviriya, P. Castells, J. Muthukumarasamy and E. Salcher. *Declarative Interface Models for User Interface Construction: The Mastermind Approach*. In "Engineering for Human-Computer Interaction", L. Bass and C. Unger, eds. Chapman & Hall, 1996, pp. 120-150.
7. A. R. Puerta. *A Model-Based Interface Development Environment*. IEEE Software, v. 14, n° 4, 1997, pp. 40-47.
8. A. Cypher, ed. *Watch What I Do: Programming by Demonstration*. The MIT Press, 1993.
9. P. Castells and P. Szekely. *Presentation Models by Example*. In "Design, Specification and Verification of Interactive Systems '99", D.J. Duke and A. Puerta, eds. Springer-Verlag, Viena, 1999, pp. 100-116.