

Comparing Graphs and Trees for Adaptive Hypermedia Authoring

*An experimental setup using the new version of the TANGOW Adaptive
Hypermedia System*

Manuel Freire ^{a,1} and Pilar Rodríguez ^a

^a *Escuela Politécnica Superior, Universidad Autónoma de Madrid, Spain*

Abstract. A key decision in designing authoring tools for Adaptive Hypermedia is the type of representation to use when authoring the relationships among contents. This paper describes an experiment that compares two representations, one based on trees, another using clustered graphs for the same task. An effort has been made to isolate all other aspects of authoring from the comparison; the choice of trees vs. graphs should be the only difference between both editors, which have exactly the same capabilities and goals: to author courses for the new version of the adaptive educational hypermedia system TANGOW. Experimental results for a small survey using the described setup are included.

Keywords. Adaptive Hypermedia, Visualization, Focus+Context, TANGOW

1. Introduction

Adaptive Hypermedia (AH) systems are based on the idea of adapting site contents to users, instead of using a one-size-fits all approach. They maintain some sort of information about the user, and the user's goals and previous actions (the *user model*[4]), and try to present content that will be relevant to the current task for that particular user by performing different types of adaptation on a (structured) content repository; the relationships among the elements in the content repository define a structure, which is often termed *content model*, or when a repository refers to a specific domain, *domain model*. In an educational context, a content repository usually consists of multiple individual courses (although this distinction can be blurred if re-usability is inserted into the equation, and several courses share contents).

The structure of an adaptive hypermedia course is therefore invariably more complex than that of static one: adaptation is performed by following certain procedures that relate a given user model with the domain model, and the domain model must include some sort of extra information that guides the adaptation process in this task. This information is usually added as meta-data, labelling content units or subunits with special tags and establishing relationships among them. An adaptation engine later makes use of this information and a user

¹Correspondence to: Manuel Freire, Av. Tomás y Valiente 11, ES-28049 Madrid, Spain.
Tel.: +34 91 4972267; Fax: +34 91 4972235; E-mail: manuel.freire@uam.es

model to present a seamless course tailored to this user's perceived goals and needs. Some relations among course units usually have much greater priority than others. For instance, the *part-of* relationship typically defines the high-level layout of a course, a layout that can then be altered before presentation by the adaptation engine (maybe *part-of-if* would be a better description), and then further refined at lower levels of locality to provide more fine-grained types of adaptation.

We present an experiment comparing different structural representations in two authoring tools developed for the WOTAN adaptive hypermedia course system. WOTAN is a new version of TANGOW[7,6], an educational AH system developed at the Universidad Autónoma de Madrid by the author's research group. In section 2, the basics of the adaptation mechanism used in WOTAN are described. The editors are described in section 3, with emphasis on the user's interface to the structure rather than the actual details involved with course update, user model expression syntax, etc. The tree-based editor is introduced in 3.1, and the graph-based editor is discussed next. In section 4, we describe an experimental setup designed to compare both approaches in different authoring tasks, and discuss the results gathered in an experiment with a small number of users.

Finally, section 5 provides concluding remarks and lines of future work, and insists on the generality of the representation problem, which we believe to be common to any adaptive hypermedia system that performs global adaptation on a closed, tightly coupled content repository.

2. WOTAN, a new version of TANGOW

The TANGOW adaptive system has been updated in the last year. The new version, WOTAN, features better integration among the administration, presentation and authoring tools, and uses XML files instead of database tables for course data, user model, and system configuration information. A new exercise subsystem has also been introduced. The remainder of this section provides an overview of the system architecture, the course model, and the user model, as a basis for the presentation of the authoring tools.

The renovation of TANGOW has been partly inspired by other adaptive hypermedia course systems, particularly by work on the AHA system [9,15] and adaptive hypermedia reference models such as [3], although in TANGOW there is no proper separation between the "teaching model" and the "content model".

2.1. Architecture

WOTAN is implemented as a set of modules (servlets inside a Java-based web application) that share session information. The `exer` module provides a web-based interface that allows authors to create and test different types of exercises, which can later be included in courses; the `admin` module implements access control and provides user and course management. Authors can upload and download courses from the system through this interface. Finally, the `pres` module is in charge of presenting courses to users, performing adaptation as needed. Currently, it is designed to closely mimic the behavior previous TANGOW interface, but the adaptation module is easily replaceable for new adaptation requirements.

Course authoring occurs client-side, and completely off-line; this allows much greater interactivity within the authoring tool, and richer representations of the course structure.

2.2. User model, course model, and adaptation

Fig. 1 illustrates the contents of a user session. Sessions include a user model with global student data (such as preferred language, learning styles[13], etc.), and a course-specific “overlay” for each course the user is enrolled in. All model information is encoded in attribute-value pairs, where the attributes of course overlays mark a location in the course model where the attribute is of relevance. If the user is currently performing a course, the current course model (as annotated by the corresponding overlay) will also be stored in the session; when the session is finished, it will be serialized back into the user model’s course overlay for that particular course.

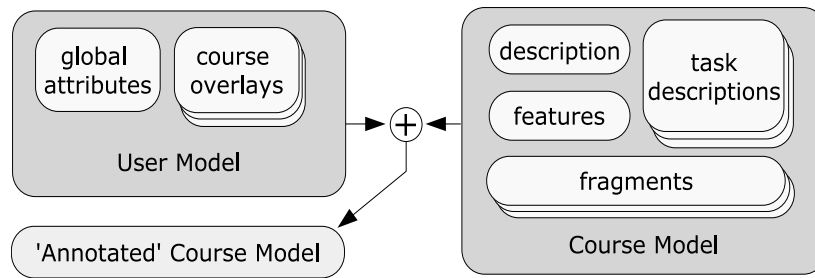


Figure 1. Construction of an annotated course model in WOTAN, the new version of TANGOW.

Adaptation data is contained in the course description, an XML document which describes a course in terms of tasks, rules, fragments, versions, and adaptation features.

Tasks are blocks of content, which may contain both other blocks and/or content fragments. A course always defines a *main task*, which acts as the entry point to the whole course. Containment is not exclusive: several tasks may refer, either directly or indirectly, to the same subtasks or fragments; but no task may include an ancestor as a subtask. Therefore, in terms of inclusion, TANGOW courses are directed, acyclical graphs (that is, graphs where no cycles can be created by following edges).

The decomposition of a task in subtasks is governed by *rules*. Each rule has an activation condition (a condition on the user model that must be met in order for the rule to be considered active), a list of subtasks, and a criteria for sequencing the subtasks (deciding in which order they will be recommended) and evaluating successful task completion. When several rules are simultaneously active for a given task, only one may be triggered (the student has a choice here). In order to simplify navigation, at most one rule for each task may be triggered. This means that, in any given moment, the path that a student has taken throughout a course will be a well-formed tree. The task + rule mechanism implements navigational adaptation in TANGOW.

Fragments are small snippets of html, images, applets, etc. A fragment may have several versions, and each version will have an associated expression that evaluates its suitability to a given user model. The most suitable version of each fragment in a task is selected when building the html pages that will represent that task to a given user, implementing canned-text adaptation.

Features describe additional aspects to be added to the user model for this particular course. For instance, in a course on microprocessor design, it may be relevant to know the user’s familiarity with boolean algebra. A feature *boolean-algebra* could be added as a “feature” to be considered in the course. Features are initialized when a course is first visited, using an automatically generated questionnaire, and the values filled in are later

available for adaptation purposes. In this case, the feature's value would be accessible as *map.course.boolean-algebra*.

Additionally, certain built-in user model characteristics (such as preferred language) are available for all courses, and the whole domain model itself (including full history) is available for adaptation. For instance, `map.task.booleanExercises.grade >= .5` (representing the final grade in task that includes a set of exercises about boolean algebra) may be better than a user-provided "feature" value when deciding whether the user really has an idea about boolean algebra. Expressions using values from the annotated user model are used in rule activation, fragment version selection, task finalization conditions, and parameter propagation (that is, pervasively) around the system, and have been implemented using the well-known Java Expression Parser package[1].

3. Editors

Authoring tools for educational AH must provide an interface suitable for authors with good knowledge of their respective domains, but not necessarily intent on an in-depth study of the system's internal workings. The interface should present an overview of the course structure, and allow direct modification of this structure as the course evolves. As described in [8], high-quality AH courses are more likely to evolve than to be completely designed from the ground off, so the interface should allow for both course creation and later maintenance.

Two obvious candidates for structural representation are trees and graphs. Trees provide a familiar top-down model of course organization, as found in the Table of Contents of many reference materials. Most AH systems use trees in their authoring tools, either exclusively or in addition to other representations (for instance, [9,5,7]). The only drawback of using trees is that non-hierarchical relationships are hard to represent inside a tree. These are, however, commonly represented by directed graphs (which are a generalization of trees where individual nodes can have more than one 'ancestor'). Graph-based tools for adaptive hypermedia editing are becoming more common, as in AHA 2.0's graph-based editor[9] and TANGOW's ATLAS editor [12]. Trees tend to be very uniform interfaces, with standard expand and collapse operations, and well-understood semantics. The extra liberty provided by graph representations makes graph-based systems much more heterogeneous.

We present two authoring tools for WOTAN courses, capable of exactly the same tasks, and which differ only in their representation of courses. In fact, both are stand-alone Java applications that share most of their code-base. Course authors are expected to download an existing course from the system as a compressed file (or create a new one with the chosen editor), edit the course locally, and upload it again to the system.

The editors also allow authors to import previous database-bound courses into the new format, and assist a user in editing and adding fragment versions. However, this shall be of no concern in the following subsections, which deal almost exclusively with the structural interface.

3.1. The tree editor

This editor, depicted in Fig. 2, is based on the interface found in a previous web-based editing tool for TANGOW. The component represents a course structure as an expandable tree, where nodes can represent tasks, rules, fragments and fragment versions. Color-coded icons are used to denote each of these elements. Fragment versions are the only real leaves.

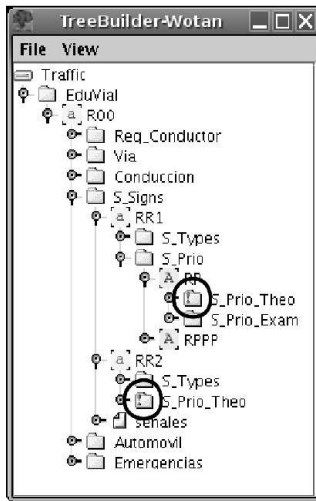


Figure 2. The tree interface, displaying a part of the “Traffic” course. The circles mark a task accessible through several paths.

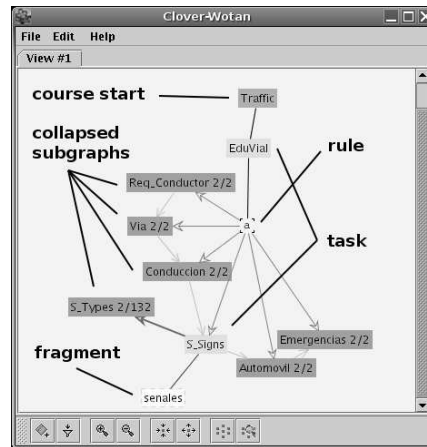


Figure 3. The graph interface, displaying the same course. Annotations have been superimposed in Arial typeface.

The tree can be expanded and collapsed as usual, contains no internal state (so that, should the course structure change due to editing, a simple revalidation will bring the representation up-to-date), and can be filtered according to a “user model” filter.

In Fig. 2, an example illustrates the main drawback of using trees for course structure representation: if a single task is reachable through different routes, it is marked with an exclamation mark (in the example, “s_prio_theo”, highlighted by an arrow). It is difficult to represent these non-hierarchical relationships in an inherently hierarchical representation such as a tree.

3.2. The graph-based editor

The graph-based editor is significantly more complex. Its initial aim was to solve the issue of non-hierarchical relationships encountered with the tree-based approach. However, the use of graphs introduces new problems: graphs are significantly less efficient in their use of screen space, require non-trivial layout to be performed (either automatically or by the user), and result incomprehensible beyond a certain size.

This has motivated the adoption of several information visualization techniques to reduce the complexity of the graphs being presented, while maintaining desirable characteristics that make them comparable to trees when trees alone would suffice. In the first place, automatic layout has been preferred to manual layout, as we understand that manual layout places too much of a burden on users, and automatic layout is both feasible and pleasant as long as the node count is kept low. This is achieved via clustering of related nodes. Navigation throughout the graph is performed by clicking on nodes, which will convert the selected node into the current ‘Point of Interest’, and expand nearby ‘interesting’ nodes and collapse distant ‘non-relevant’ ones. The default behavior of this fisheye-lens[11] (or focus+context) approach can be modified by either freezing certain nodes (so that neither their position nor their collapsed-uncollapsed status will be altered by visualization changes) or modification of the focus variables: number of nodes to be shown, and size of the neighborhood to expand.

Fig. 3 is a screen-shot of the graph interface. Color-coding and shapes are used to distinguish different node and edge types. The edge between “S_Signs” and “S_Types” is a

“cluster edge”, that is, contains several edges (see the tree representation of the same course, in Fig. 2, to identify the edges involved).

Animation of transitions is used in order to preserve the user’s mental map of the course during navigation, as every change in viewpoint or filtering implies another layout of affected nodes, and suddenly changing node positions tend to be annoying.

The graph-based editor is built on top of CLOVER [10] (Cluster Oriented Visualization EnVironment), a graph visualization framework that uses the JGraph[2] component. CLOVER is not specific to TANGOW courses, or indeed adaptive hypermedia; it has been used for graph visualization in other application domains, such as the reusable content creation system Targeteam[14,10].

4. Experimental Setup

As both editors share the same editing capabilities, the goal of the experiment is to measure author speed, accuracy and satisfaction with each of the editors. First, experiment participants receive a brief primer to WOTAN course structure and an introduction to each of the tools and a description of the tasks to be performed. They are also presented with a printed page with the symbols used in course representation and a list of requested tasks. Finally, the participant is asked to perform the tasks on an example course, in order, first using one editor and then the other. The order of tool selection is changed for each participant, so that half of the participants start with the tree-based tools, and the other half starts with the graph-based one. As the same tasks are performed twice, once for each tool, by users without previous experience with the system (or AH in general), we expected to find the speed and accuracy of the second run to be better than those of the first run.

4.1. Tasks

All tasks are performed on the example course depicted in the previous tool screenshots. The tasks are designed to be simple but representative of a typical editing session, and are presented in order of increasing difficulty. For each task, the time required to complete it successfully (as determined by the experimenter upon request) is recorded. After all tasks for both tools have been completed, the user is requested to fill in a small "difficulty survey" assessing the difficulty of each task/tool combination.

- Task 1: find and change the name of a given fragment node, identified by a path that leads to it. Subjects will have to navigate using the corresponding tool, which is the only problem presented by this task.
- Task 2: locate all paths leading to a task node. One of the paths is provided in the statement. This task is representative of a typical problem in AH courses, and we expected the graph representation to perform better than the tree representation.
- Task 3: add a rule that will be active only for users with a specific profile (rule location and profile-matching information are provided in the statement), insert a task into this rule, and add a fragment to the inserted task. This tests basic course growth.

4.2. Results

Experiment results can be found in figures Figs. 4 and 5. Only 8 participants were involved, divided into two groups. Charts on the top row represent time in seconds required to perform

each task. In each chart, there are 6 blocks of columns; the first three columns represent the time required to perform each of the 3 tasks with each of the editors, in order, from left to right. Charts in the second row display perceived difficulty of each task with each editor (filled in after performing all tasks with all editors), adjusted so that the least difficult task for each user begins at “difficulty level” 1.

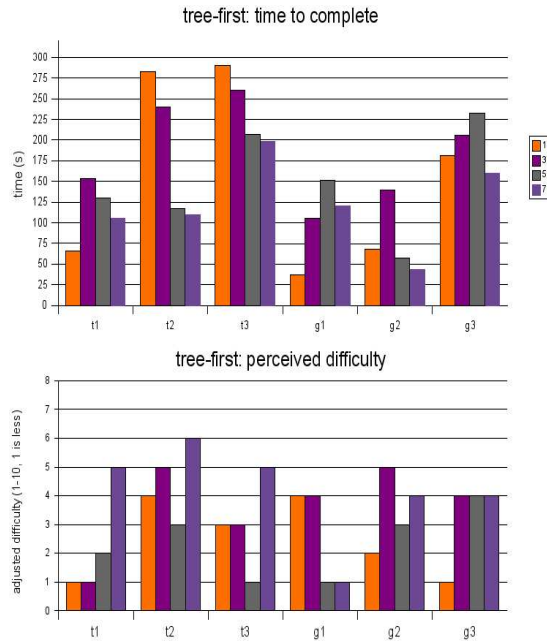


Figure 4. Participants that started with the tree-based editor. Tasks on the *left* were performed first.

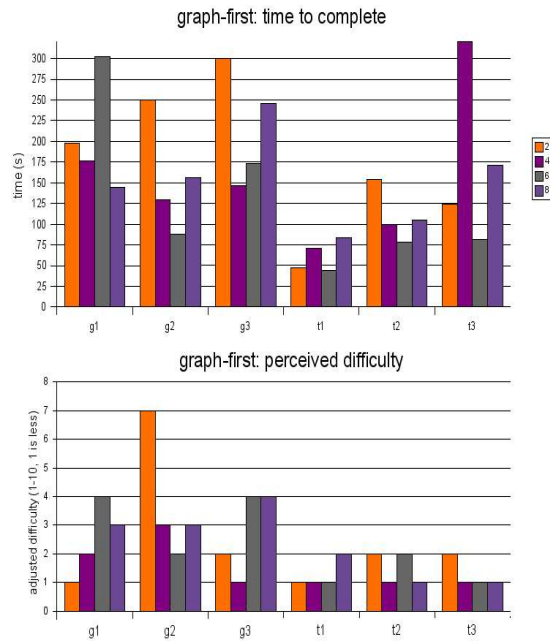


Figure 5. Charts for participants that started out with the graph-based editor.

Statistical analysis is not meaningful with such a small sample, but certain trends are visible. For instance, task times are much shorter in the second run than in the first run, regardless of the editor that was used first. This suggests that familiarity with the system’s concepts and the course structure is harder to master than interaction with the editors (users had only received a very basic introduction to both).

Task times with each group’s first tool are very similar, although Task 1 seems to take longer when performed for the first time with the graph editor, probably because graph navigation is more complex than tree navigation. As expected, Task 2 is performed slightly faster when using the graph editor. Better initial training should have lowered times much more. Perceived difficulty is also usually lower for Task 2 when using the graph-based editor. It is also interesting that users rate tasks as more difficult when they have started out with the tree editor. This suggests that those using the graph-based editor had a better understanding of the course structure on their second run, and found tasks easier to accomplish.

In general, despite the differences in interface, both tools are roughly equal for non-experienced users when performing simple tasks. The graph tool is perceived as harder to use and takes longer to get used to than the tree tool, but seems to provide better understanding of course layout. Familiarity with the system is gained rapidly, but still accounts for most of the difference between both runs.

5. Concluding Remarks and Future Work

Adaptive Educational Hypermedia authoring tools must choose a suitable interface to present content structure and relationships. The two main candidates are tree-based and graph-based representations. An experimental setup to compare two approaches to course structure representation, one using trees and the other using graphs, is presented. The main feature is that both tools share exactly the same capabilities, differing only in course representation. Although the authoring tools under comparison are both designed with the WOTAN system in mind, we believe that the ideas behind this experiment are applicable to a broad range of adaptive hypermedia systems.

Experimental results suggest that graph-based representations, although harder to master, are better suited to navigation and orientation in AH structures than tree-based ones. However, the evidence is inconclusive: the sample is small, and familiarity with the system and the requested tasks should be factored out of the experiment to provide a more meaningful comparison. A follow-up experiment with a larger sample size is planned for the near future, including a longer introduction to the WOTAN system and both editors, and a wider array of tasks. Interface glitches discovered during this first experiment will be fixed in time for the follow-up.

Acknowledgments

This work has been sponsored by the Spanish Ministry of Science and Technology (MCyT) with project code TIN2004-03140.

References

- [1] Java expression parser. <http://www.singularsys.com/jep/>.
- [2] Jgraph java graph visualization component. <http://www.jgraph.com/>.
- [3] Paul De Bra, Geert-Jan Houben, and Hongjing Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In *UK Conference on Hypertext*, pages 147–156, 1999.
- [4] P Brusilovsky. Adaptive Hypermedia. In *User Modeling and User-Adapted Interaction*, volume 11, pages 87–110. Kluwer Academic Publishers, The Netherlands, 2001.
- [5] Peter Brusilovsky and Gerhard Weber. Elm-art: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education*, 12:351–384, 2001.
- [6] R. M. Carro, E. Pulido, and P. Rodriguez. Dynamic generation of adaptive Internet-based courses. *Journal of Network and Computer Applications*, 22(4):249–257, October 1999.
- [7] R. M. Carro, E. Pulido, and P. Rodriguez. TANGOW: a Model for Internet Based Learning. *International Journal on Continuing Education and Life-Long Learning*, 11(1–2), 2001.
- [8] Alexandra Cristea and Lora Aroyo. Adaptive authoring of adaptive educational hypermedia. *Lecture Notes in Computer Science*, 2347:122–132, 2002.
- [9] P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! Version 2.0, More Adaptation Flexibility for Authors. In *Proceedings of the AACE ELearn'2002 conference*, pages 240–246, October 2002.
- [10] Manuel Freire and Pilar Rodriguez. A graph-based interface to complex hypermedia structure visualization. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 163–166, New York, NY, USA, 2004. ACM Press.
- [11] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23. ACM Press, 1986.

- [12] J. A. Macías and P. Castells. Interactive Design of Adaptive Courses. In M. Ortega and J. Bravo, editors, *Computers and Education - towards an Interconnected Society*. Kluwer Academic Publishers, Dordrecht (The Netherlands), 2001.
- [13] Pedro Paredes and Pilar Rodriguez. Considering sensitive-intuitive dimension to exposition-exemplification in adaptive sequencing. *Lecture Notes in Computer Science*, 2347:556–559, 2002.
- [14] J. Schlichter and G. Teege. Web-based information management for university education. In Jari Multisilta and Jari-Pekka Niemi, editors, *Proceedings of LETTeT 98 and MaTILDA 98 joint conference*, pages 99–106, May 1998.
- [15] N. Stash and P. De Bra. Building Adaptive Presentations with AHA! 2.0. In *Proceedings of the PEG Conference*, July 2003.