A Graph-Based Monitoring Tool for Adaptive Hypermedia Course Systems

Manuel Freire and Pilar Rodríguez

EPS-Universidad Autónoma de Madrid, Madrid ES-28049, Spain,
{manuel.freire,pilar.rodriguez}@uam.es,
WWW home page: http://www.ii.uam.es/~mfreire

Abstract. Adaptive hypermedia courses are difficult to debug, validate and maintain. Logfile analysis is partly to blame. We propose a graph-based approach to both real-time student monitoring and logfile analysis. Students are represented at their current locations in a dynamically created map of the course. Selected parts of student user models are visually exposed, and more detail is available on demand. Hierarchically clustered graphs, automatic layout and focus+context techniques are used to keep visual complexity at a manageable level. This component has been developed for an existing AH course system. However we believe that our approach can be readily extended to a wide selection of adaptive hypermedia course systems, filling in an important gap during course creation and maintenance.

1 Introduction

Adaptive Hypermedia is an obvious choice for online courses. Adapting the course to the student surely sounds better than a one-size-fits-all approach. But a vast majority of online courses are not adaptive, because adaptation is difficult to design, test and maintain [1]. It is not easy to predict what users will find during browsing, and for online systems, direct observation is not an option. We propose the use of a graph-based interface to monitor and analyze course usage as an important aid to tutors and course authors.

2 Approach

The following figures illustrate our monitoring interface, which is tracking 5 users throughout a sample course. Users Alice and Bob are both in T2, while Carol, David and Edward are further along the course. In fig. 1, the current map is shown, with exactly enough detail to see the active users. In our visualization, courses are represented as maps, and particular nodes represent either specific activities (*tasks*) or decision points (*rules*). Most of the course is hidden, collapsed under the darkened task nodes marked with an asterisk. Students are represented as additional nodes connected to the last task they have accessed.

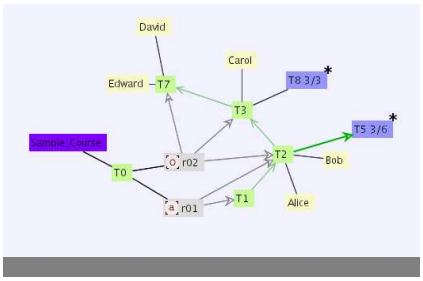

Fig. 1. Example course during monitoring; asterisks (*) represent clustered nodes.

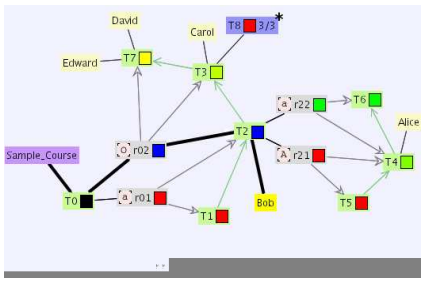

Fig. 2. Alice has advanced into T4; Bob is selected and his path becomes highlighted.

Tracking user progress is as simple as watching student nodes move throughout the map, with further details available on demand. In fig. 2, Alice has advanced a bit further, and the map has been automatically expanded to display her progress. Additionally, Bob's node has been selected: this causes Bob's past path to be highlighted. Color-coded squares have appeared next to the labels of each task, rule and cluster, indicating which parts of the course are, according to the system, recommended, available, or disencouraged.

2.1 Interaction

The interface is built on top of the CLOVER framework [2], and inherits CLOVER's implementation of hierarchically clustered graphs [3]. Clicking on a node selects it and marks that node as the current *point of interest* (PoI). Nodes near to this PoI will be presented in full detail, while nodes further away can be collapsed into clusters to keep the level of visual complexity within the desired bounds. Whenever the set of visible nodes changes, automatic layout is performed, and the old view is smoothly animated to transform into the new one, in an effort to preserve the users mental map [4]. PoI focusing and automatic layout can be toggled on and off, allowing nodes to be moved around to suit the user (marquee selection and *Ctrl+click* are also available). Both manually modified and automatic layouts are stored in a layout cache, and are later reused if the same view is revisited.

Selecting individual tasks and user nodes triggers additional visual cues. When nothing is currently selected, the map simply tracks user position. Selecting a course task annotates the users that are currently eligible to perform it, using a color-coded scheme: red for "not allowed", blue for "started", black for "finished", and a color from yellow to green to indicate a "degree of recommendation" for all other cases. Conversely, selecting a student annotates all currently-visible tasks with the same color-coding scheme. In addition, the path that this student has followed throughout the course is highlighted via increased thickness edges that have been transversed. Our system does not constrain users to following recommended links, and therefore the actual path can fall outside

existing edges. An artificial, faded path is used for these cases, connecting the unexpectedly-visited task to the nearest completed task (the course graph is guaranteed to be connected, so this is always possible). Finally, double-clicking on a student node displays a dialog with the student's current user model, color-coded again to reflect the latest changes.

Popups are available when hovering over a node, without need to select it. Currently, they have only been defined for student nodes, displaying the tasks they have started but not completed, along with a percentage indicating degree of completion and their current "grade" for that task. Since in our system tasks form a hierarchy and the whole course is subsumed under a single task, this is displayed as a tree and allows a quick assessment of the student's performance.

2.2 Events

Changes to the graph are driven by *events*. A task change, an exercise submission or a new student logging in all trigger events. Once it is received by the monitor, it translated into a series of transformations on the graph, which can then be rendered on the interface. Undo support is also present, allowing both forward and backward event navigation.

Two event sources are currently available: a real-time source that monitors a running course on our AH system, and an offline source that reads its events off a log file generated during system execution. Events are not self-contained; instead, they only carry enough information so that the monitoring tool can replay them locally on copies of all user models involved, using the same adaptation engine found in the system itself, arriving at the same changes to these models. This results in smaller logs and an efficient monitoring protocol, at the expense of a heavier monitoring tool and the burden of synchronizing user models before a monitoring session. However, we believe that the benefits greatly outweight the drawbacks: full user models are available at the monitor, and it is free to inspect the internal state of the adaptation engine. Events can be delivered preserving their time of occurrence, or issued at a constant pace (only if produced from non-realtime source types). The latter is the default for logfile playback. In logfile playback mode, additional controls have been implemented to "pause", "play" and "reverse" the event stream. Further video-like playback control is still work-in-progress.

2.3 Implementation

Our work is centered on the WOTAN adaptive hypermedia course system, a new version of TANGOW[5][6]. WOTAN courses are created and maintained with a graph-based authoring tool [7], which has been recently extended to allow course monitoring and logfile analysis; essentially the same interface is used for authoring and monitoring, with different hints and user interaction in each mode. Both the authoring/monitoring tool and the WOTAN AH course system have been implemented entirely in Java, and a large portion of the codebase is shared.

3 Concluding remarks and future work

We have implemented a graph-based monitoring tool for the WOTAN system. Although our tool is heavily geared for use with this system, this integration only provides added value (in our case, full access to user models and system state, while keeping an efficient monitoring protocol); it is by no means necessary. Graph-based interfaces can be used to monitor usage of almost any hypermedia application, and are specially suited if complex user models are in use, because of the variety of visual hints that can be presented to users.

Work is ongoing in several areas, including new event sources (such as simulated, "random" students for course stress-testing), better event stream control (video-like positioning and playback control), and mental map preservation issues when many students are logging in and out at widely separated parts of a course.

An interesting idea is to integrate tutoring into the tool, allowing tutors monitoring a course to open instant-messaging sessions with students that appear to be stuck, maybe even presenting aid requests as small notes directly on the interface. Student collaboration could also use a simplified monitoring interface to stay aware of each other's virtual location.

4 Acknowledgments

This work has been sponsored by the Spanish Ministry of Science with project code TIN2004-03140.

References

1. Cristea, A., Aroyo, L.: Adaptive authoring of adaptive educational hypermedia. Lecture Notes in Computer Science **2347** (2002) 122–132
2. Freire, M., Rodriguez, P.: A graph-based interface to complex hypermedia structure visualization. In: Proceedings of AVI'04, ACM Press (2004) 163–166
3. Eades, P., Huang, M.L.: Navigating clustered graphs using force-directed methods. J. Graph Algorithms and Applications: Special Issue on Selected Papers from 1998 Symp. Graph Drawing **4**(3) (2000) 157–181
4. Eades, P., Wei Lai, Misue, K., Sugiyama, K.: Layout adjustment and the mental map. Journal of Visual Languages and Computing **6** (1995) 183–210
5. Carro, R.M., Pulido, E., Rodriguez, P.: Dynamic generation of adaptive Internet-based courses. Journal of Network and Computer Applications **22**(4) (1999) 249–257
6. Carro, R.M., Pulido, E., Rodriguez, P.: TANGOW: a Model for Internet Based Learning. International Journal on Continuing Education and Life-Long Learning **11**(1–2) (2001)
7. Freire, M., Rodriguez, P.: Comparing graphs and trees for adaptive hypermedia authoring. In: Proceedings of the 3rd International Workshop on A3EH, AIED'05 (2005) 4–12