Generating Reports of Graphical Modelling Processes for Authoring and Presentation

Lars BOLLEN
University of Duisburg-Essen
Faculty of Engineering
Institute for Computer Science and Interactive Systems
47048 Duisburg, Germany

Abstract. Today's computer supported modelling environments could provide much more information about the users' actions and problem solving processes than they usually store for later usage. Thus, relevant information about learning processes which could be used for reflection and analysis is lost.
This paper describes an approach to tackle this issue by generating "reports", in the sense of augmented summaries of states and action traces from modelling processes. This approach includes a) gathering information about actions and states from specific modelling environments, b) analysing these information using domain knowledge (if available) and c) represent the results in a way suited for various use cases like authoring, presentations or monitoring of learning processes.

1. Introduction and problem description

In many areas of teaching and learning (especially in natural sciences and computer science), the task of modelling is crucial for students to get deeper insights into the problem domain, to be able to create hypotheses and predictions of complex phenomena and to improve communication and coordination between peers and with the teacher [1]. Especially when learners create models themselves that are executable and actively explorable, there is definitely value added compared to static representations (e.g. in books or on a chalkboard). While modelling, the learner interacts with computational objects, manipulates them and thereby makes his thoughts explicit. In this context, the phrase "objects to think/work with" has been introduced in [2], meaning that the exploration, manipulation and creation of artefacts support in establishing understanding.

Having these ideas in mind, the modelling environment Cool Modes (Collaborative Open Learning and Modelling System) [3] has been developed. Cool Modes is a framework for collaborative modelling with graph based visual languages like Petri Nets, System Dynamics, UML class diagrams and many more.

Nevertheless, when a learner finishes a modelling task with a modelling environment like Cool Modes, usually only a result is stored. The process of creating and exploring a model is compressed to a single artefact. In more detail:

- *The process* of his work gets lost, since only the result is stored. The single actions that lead to this result, are usually lost.
- *Information about different phases* (e.g. phases of argumentation and coordination with peer, design, verification, revision, annotation etc.) the user went through in his problem solving process gets lost.
- *The design rationale* is lost, unless the user made it explicit in his solution or in additional documents.

- Elaboration of *alternative solutions* usually cannot be reproduced, since older creations are simply overwritten by newer ones.
- Information about *collaboration* gets lost having only a single artefact as the output of a modelling process.

Knowledge about these issues is helpful for various target groups and for various purposes:

- The learner himself could use this information for self reflection, self / peer assessment, peer authoring / vicarious learning [4] and for presenting own results.
- Teachers could be supported in assessment, authoring (by demonstrating solutions and presenting prepared material) and for finding typical problems in students' solutions.
- Researchers in the field of AIED / CSCL could use the additional information for interpreting and understanding learners' actions and results and for applying different analysis mechanisms on the stored states and action traces.

So, having the various information mentioned above might be helpful for different target groups and different use cases. The challenge is to obtain the required information, to interpret the information, to organise and present it in a general but still useful way.

2. Related work

2.1 Record and Replay

Some approaches like Ottmann's "Authoring on the fly" [5] and Rojas' "E-Chalk" [6] store process-related data in additional to the result, too, but they use a "record and replay" approach. The purpose is to record whole lectures in universities in order to stream them via internet or to replay them later. The target group of this approach is mainly students. Only a linear structure of material is supported; alternative solutions cannot be represented easily without recording a completely new session. During the record, there is no analysis or interpretation of actions taking place. Thus, you have to cut or skip irrelevant phases manually.

2.2 Series of snapshots: COPRET

A different approach called COPRET ("COllaboration Progress REproduction Tool") can be found in [7]. Here, a collaborative discussion support tool is observed by an analysis component that generates snapshots (images) of the learning environment at well defined points in time (e.g. after a change of control between the users or after insertion, modification or deletion actions). As a result, this tool produces a Word file that contains the teacher's and the students' actions and messages as well as screenshots in a chronological order. This approach combines some basic aspects of analysing user actions and storing process-related data, but focuses on supporting teachers assessing and interpreting students' results and cannot be used for authoring or presentations.

2.3 Behavior Recorder / CTAT

Another approach that focuses on analysing and tutoring problem solving processes is described by Koedinger et. al [8]. For well-defined problems (including a well-defined user interface) like the addition of two fractions, the so-called Behavior Recorder is able to record various paths of actions, which are specified by a teacher, that lead to correct or incorrect solutions. Enriching these paths manually with tutor messages builds a pseudo cognitive tutor that is able to feed back messages into the learning environment when the same (correct or incorrect) actions are done again by a learner. Recently, this approach has been extended to

analyse collaborative, open modelling tasks [9]. Alternative solutions can be recorded and displayed, but they cannot be fed back into the modelling environment. Thus, the focus in this approach is on analysing and tutoring users' actions.

3. Approach and prototypical implementation

The problem that has been described in the chapters above can be addressed and solved by generating *reports*. Reports, in the sense of this approach, are summaries of states and action traces from modelling processes. The problem description raises some requirements that a report generation tool has to comply with.

Generally speaking, a modelling environment can provide information about the actual *state* of a model as well as information about the *actions* that the learners execute while modelling. Thus, a report generation tool has to be able to gather and organise both types of information. Collecting information about *states* and about *actions* of a modelling environment enables for a rich examinations of the modelling process, since action-based analysis methods as well as state-based analysis methods can be applied [10].

Having these different kinds of information calls for an appropriate way of visualising them. A graph-based visualisation of a report generation tools seems to be adequate for two reasons. First, most modelling environments use graph-based visualisations themselves, so it reduces cognitive load when teachers, students or researchers are using a modelling environment and a report generation tool. Second, when talking about traces of actions that lead to different states, it seems to be quite similar to paths on a map that lead to different places. Thus, a graph-based representation seems to be quite naturally (see figure 1).

The problem description came up with the possibility of using reports for presenting results to peers or to students. This requirement can be fulfilled by having means for interactive browsing and for feeding states from the reporting tool back into the modelling environment, thus providing *flexibility in presentation*.

When using a report generation for authoring learning material, re-arranging, modifying and establishing new connections between states in the captured material has to be possible, providing *creativity in authoring*.

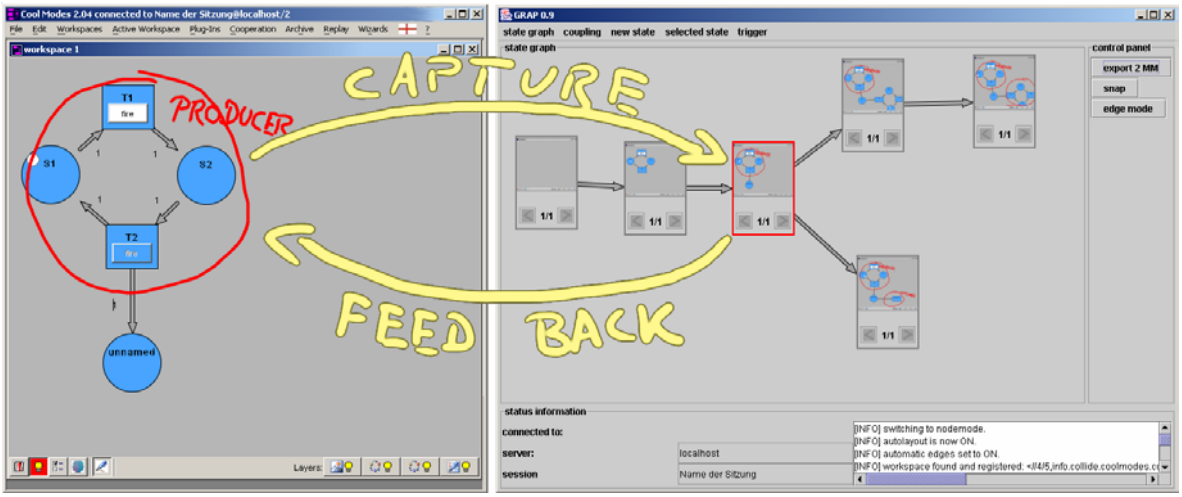

Figure 1. Using GRAP with Cool Modes when modelling a Petri Net. On the left, the modelling environment Cool Modes is shown; on the right, the report generation tool GRAP. The basic modes of operation, capturing and feeding back, are sketched in the figure.

On the way fulfilling and implementing these requirements, a prototypical report generation tool called "GRAP" ("Generating Reports for Authoring and Presentations") has been created. GRAP's basic modes of operation are capturing states of a modelling

environment at specific times during the learning / modelling process as well as storing the actions that occur between these states, display this information in a graph-based structure and feeding these states back into the learning support environment. Typically, the nodes of this graph represent the states of a model while the edges hold the information about the users' actions (see figure 1).

The decision about suitable moments for capturing is critical for having a useful summaries of modelling processes. According to specific usage scenarios, this can be decided by the user of GRAP herself (e.g. for authoring, see chapter 5) or automatically by the system (e.g. for automated documentation). In the latter case, the report generator has to interpret the actions to be able to detect phases or milestones in the modelling process to find detect suitable moments for capturing. This interpretation of user actions is usually dependent on the domain that these actions are related to, as it has been described in [1].

Cool Modes provides all requested features like supplying information about the actual state of the model, about the users' actions and it is capable of playing back states and actions. Technically speaking, Cool Modes is able to be synchronised with other application by using a communication server called MatchMaker [11], replicating the state of the modelling environment on each client and propagating user actions. GRAP is attached to this distributed system as just another client, able to listen to user actions, to capture the complete state of the modelling environment and to feed information (e.g. states or actions) back into the system.

These features made Cool Modes an appropriate candidate as a modelling environment to build a prototypical implementation of a report generation tool. At the current moment, GRAP is able to capture the states and user actions of the modelling environment Cool Modes at any time. This information is organized in a graph-based structure (see figure 1, right hand side); the nodes represent different states, the edges represent the actions that lead from one state to another (these actions are not shown in figure 1). The states can be fed back into the modelling environment at any time.

4. Scenarios

To clarify this approach and to point out the usefullness of the solution described above, three scenarios will be presented in the following.

4.1 Authoring

A teacher of a biology course in school chooses the to model predator-prey interactions (e.g. foxes and rabbits) as a topic for the next lesson. He decides to use a modelling software like Cool Modes (using the System Dynamics plug-in [12]) to model, visualise and simulate the equations that represent the predator-prey interactions. Creating the model from scratch would be too time consuming for a lesson in school; presenting a pre-constructed model is probably too difficult for the students. He starts creating the model the day before at home, using GRAP to snap important steps during the modelling process. He creates several, different, non-linear *presentation paths* to be able to explain intermediate results and alternatives, to show typical mistakes and dead ends and to have answers to possible questions at hand. During the lesson, the teacher uses GRAP to feed back different stages of the model into the modelling software, still being able to use the modelling tool as such, rather than doing a predefined slide show. GRAP may propose *presentation paths* from given start and end states to visualise (and to understand) the evolution of a desired solution.

Thus, GRAP would be used in a way similar to an authoring tool. Prepared material can be arranged in a complex, non-linear structure to have flexible means for presenting prepared material, still being able to shift into modelling activities and simulations at any time.

4.2 Documentation on-the-fly

During discussion in class at school or during meetings in a research group, it is quite common to create concept maps of the problem domain, to create QOC [13] networks to document design decisions or to point out (or even solve) problems. For people that could not attend the meeting, for students trying to remember the course of a lesson, or simply for late-comers, it is often difficult to reconstruct the meaning and creation process of a model or the evolution of a concept map. GRAP will be able to analyse modelling actions while they take place, decide about important stages and take snapshots from these important steps. These snapshots are arranged in a graph, showing which state lead to another. In a linear process this graph will be a simple linked list. However, if the users go back to a previous state and continue from there to create an alternative solution, the graph becomes more complex. This graph can be used to catch up on the content of the discussion, meeting or lesson.

In this way, GRAP can be used for documenting and retrieving various modelling processes. Viewers do not have to watch whole replays, but can fall back on a compressed, yet relevant summary of a modelling process.

4.3 Monitoring and analysing

A researcher in the field of computer-supported learning and artificial intelligence in education is interested in particular features of collaboration such as joint exploration of a given model. He uses GRAP to capture the state of a modelling environment at specific times while groups of students are trying to solve modelling tasks. Various filters and analysis methods could be applied to the states and action traces that are gathered by GRAP. Even more, results and process related data from different groups of students are displayed and compared at the same time, still being able to feed back intermediate results to get a detailed insight into the learning process that is being analysed.

Here, GRAP is used to support analysis of learning processes by capturing states and action traces from modelling environments and applying state based analysis methods as well as action based analysis methods.

5. Challenges and future work

On the way to finish this prototypical implementation, several challenges have to be mastered:

One of these challenges is an appropriate analysis, interpretation and assessment of user actions, potentially taking into account the domain and the context of these actions. This interpretation of user actions helps identifying phases and milestones in modelling processes, thus being a way to find suitable moments for capturing the state of a model.

Another challenge is the elaboration of the meaning of edges between states in the report graph. Here, an appropriate approach for teacher's authoring could be the classification into didactical relations (e.g. "X introduces to Y"; "X is exemplified by Y") as described in [14].

Another interpretation of the meaning of edges might be the "degree of relevance" of user actions. Considering the domain and context of an action (as described above), the system will be able to decide the relevance of particular actions for the modelling actions.

A challenge on a broader scale is to find a way to integrate a report generation tool like GRAP into several other modelling tools like CoLab [15] or ModellingSpace [16]. Therefore, a common data format to describe actions and states on modelling environments has to be found or defined. A standardised message and state description opens the way for integration, which would be obviously beneficial.

6. Conclusion

It has been described, that the potential information about learning and modelling processes from modelling environments is not used to a full extent at the moment. Details about intermediate states of models and about users' actions get lost in most cases.

This information can be valuable for target groups like students, teachers and researchers for use cases like authoring, documenting and analysing modelling processes.

An approach and a prototypical implementation called GRAP has been described. This approach uses state based and action based analysis to capture the states of a modelling environment at specific time as well as storing the users' action that occurred. The captured states and actions are summarised in a report, that is visualised in a graph-based structure. These states can be fed back into the modelling environment.

Still, there are several challenges like using domain knowledge to interpret the actions, calculating suitable moments for automated capturing of states and finding a common, standardised description for states and action of modelling processes.

7. References

[1] Harrer, A., and Bollen, L. (2004). *Klassifizierung und Analyse von Aktionen in Modellierungswerkzeugen zur Lernerunterstützung.* In Workshop-Proc. of Modellierung 2004, Marburg.

[2] Harel, I. and Papert, S. (eds.) (1991): *Constructionism*. Ablex Publishing. Norwood, NJ.

[3] Pinkwart, N. (2003). *A Plug-In Architecture for Graph Based Collaborative Modelling Systems.* In Proc. of the 11th Conference on Artificial Intelligence in Education (AIED 2003), Amsterdam, IOS Press.

[4] Rummel, N., and Spada, H. (2002). *Combining worked-out examples and vicarious learning to promote the coordination of computer-mediated interdisciplinary collaboration.* In Proc. of AERA 2002, New Orleans.

[5] Müller, R., Ottmann, T. (2000). *The "Authoring on the Fly" System for Automated Recording and Replay of (Tele)presentations.* Special Issue of Multimedia Systems Journal, Vol. 8, No. 3, ACM/Springer.

[6] Rojas, R., Knipping, L., Raffel, W.U., and Friedland, G. (2001). *Elektronische Kreide: Eine Java-Multimedia-Tafel für den Präsenz- und Fernunterricht.* In Beck, Sommer (eds.): Tagungsband der Learntec, Vol 2, pp. 533-539, Karlsruhe.

[7] University of Aegean / LTEE (2004). *State of the Art on Interaction Analysis: Interaction Analysis Indicators*. Deliverable D.26.1 of Kaleidoscope "Network of Excellence" research project on "Interaction and Collaboration Analysis supporting Teachers' and Students' Self-regulation".

[8] Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. M., and Hockenberry, M. (2004). *Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration.* In Proceedings of 7th International Conference on Intelligent Tutoring Systems, ITS 2004, Maceio, Brazil.

[9] McLaren, B., Bollen, L., Walker, E., Harrer, A., and Sewall, J. (2005). *Cognitive Tutoring of Collaboration: Development and Empirical Steps Towards Realization.* Accepted for the Conference on Computer Supported Collaborative Learning, CSCL 2005, to take place in Taipei, Taiwan in May/June 2005.

[10] Gaßner, K., Jansen, M., Harrer, A., Herrmann, K., and Hoppe, H.U. (2003). *Analysis methods for collaborative models and activities.* In Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL 2003), Bergen.

[11] Jansen, M. (2003*). Matchmaker - a framework to support collaborative java applications.* In the Proceedings of Artificial Intelligence in Education (AIED 2003), IOS Press, Amsterdam.

[12] Bollen, L., Hoppe, H.U., Milrad, M., and Pinkwart, N. (2002). Collaborative Modelling in Group Learning Environments. In Proc. of the Int. Conf. of the System Dynamics Society, Palermo (Italy), July 2002, pp. 53.

[13] MacLean, A., Young, R., Belloti, V., and Moran, T. (1991). Questions, options, and criteria: Elements of design space analysis. Human-Comput. Interaction, 6(3–4), 201–250.

[14] Baloian, N. (1997). Strukturierte Erstellung und Kooperative Nutzung von Instruktionsmaterial in einem Computerintegrierten Klassenraum. PhD Thesis. University Duisburg, Germany, September 1997.

[15] CoLab – Collaborative Laboratories for Europe, website http://www.co-lab.nl

[16] ModellingSpace - website http://www.modellingspace.net