From "Events" to "Activities": Creating Abstraction Techniques for Mining Students' Model-Based Inquiry Processes

Vilaythong Southavilay*, Lina Markauskaite**, Michael J. Jacobson**

*School of Information Technologies
University of Sydney
vstoto@it.usyd.edu.au

**Centre for Research on Computer Supported Collaborative Learning and Cognition
University of Sydney
lina.markauskaite@sydney.edu.au
michael.jacobson@sydney.edu.au

ABSTRACT

In this paper, we present a technique that we have developed to transform sequences of technical events into more abstract actions and semantic activities. The sequences of more abstract units are then used for discovering patterns of students' interaction with computer models using heuristic miner. Our proposed approach automatically segments sequences of technical events that occurred during model runs and pauses and, on the basis of the nature of technical events that occurred during model runs and pauses, clusters them into actions. Then, using heuristic rules, it classifies actions into activities. We demonstrate the usefulness of our multilevel abstraction for extracting and exploring characteristic patterns of students' interaction with computer models. Our study shows that each abstraction level could help to identify distinct characteristics of students' interaction.

Keywords

Process mining, data pre-processing, multilevel data analysis, model-based learning.

1. INTRODUCTION

It has been acknowledged that model-based inquiry could be a very effective pedagogical approach for teaching and learning complex scientific knowledge. Computer models could help students to engage in first-hand scientific investigations of complex social and natural phenomena and construct deep, authentic understanding of many complex processes, such as climate change [3, 4]. However, a number of studies that investigated model-based learning have found that not all learners succeed achieving desired learning outcomes [1, 8, 9]. They suggested that students' failure and success to learn from computer models may be related to the differences in how students interact with computer models. For example, studies demonstrated that some students, when they explore computer models, systematically change one parameter after parameter; in contrast, other students approach tasks in more haphazard ways and change different model settings simultaneously [6]. While the former students usually succeed completing given inquiry tasks, the latter students tend to be less successful demonstrating desired learning outcomes. However, methods for exploring students' model-based inquiry processes have been complicated, usually based on human coding, thus hard to implement in computer systems. Only recently researchers have started to explore the possibilities to extract students' inquiry characteristics and patterns automatically from the log files [2, 7]. These techniques could help to detect productive and unproductive students' behaviours and scaffold students' inquiry automatically.

A traditional approach in exploring learning processes automatically is to use event logs of students' interaction with computer software as an input to process mining algorithms. However, processes of students' interaction with computer models tend to be very flexible, unstructured and composed from large numbers of fine-grained technical events. Consequentially, technical events may be too far remote from students' intended purposeful actions and the identified patterns could be hard to understand and use.

Our goal is to develop a method for identifying semantically more abstract and meaningful students' actions and activities from lower level technical events recorded in the logs. The sequences of more abstract units then could be used to investigate students' model-based inquiry patterns. In this paper, we present a multilevel data preprocessing approach that we have created and used in combination with process mining for investigating students' model-based inquiry strategies. Using real data, we illustrate that by identifying more abstract semantic units and examining their sequences we can gain additional insights into how students interact with NetLogo computer models.

The rest of this paper is organised as follows. Section 2 reviews related work and introduces our approach. Section 3 illustrates the capability of our approach using real event log data. Section 4 presents conclusions.

2. RELATED WORK AND APPROACH

There have been a number of studies that focused on analyzing data logs of less-structured processes. However, the process models extracted from raw data logs using traditional process mining techniques have been usually difficult to interpret. One of recently proposed techniques to deal with this issue is abstraction. Particularly, Bose and Aalst [1] explored a way that identifies the looping constructs in logs and replaces the repeated occurrences of the loops by more abstract entities (aka. activities). They also proposed a technique for discovering sub-processes or common functionalities in the traces and replacing them with more abstract entities. Our technique adopts Bose and Aalst [1] approach and focuses on how event logs could be semantically transformed into more abstract action and activity sequences. However, it interprets looping events and repeated occurrences of events differently from Bose and Aalst [1]. Our technique segments technical event-sequences into "bags of events" and then uses a heuristic-based classifier to automatically identify activities.

2.1 Study Context and Data

Data for this study came from a larger design-based project that investigates how school students learn scientific knowledge about

climate change [4]. In total, 90 students from a high girls' school learnt about climate change by exploring computer models. Students completed inquiry tasks in dyads. In this paper, we conducted the analysis of dyads' interactions with the Carbon Cycle model (Figure 1). In total, we recorded 21 event log.

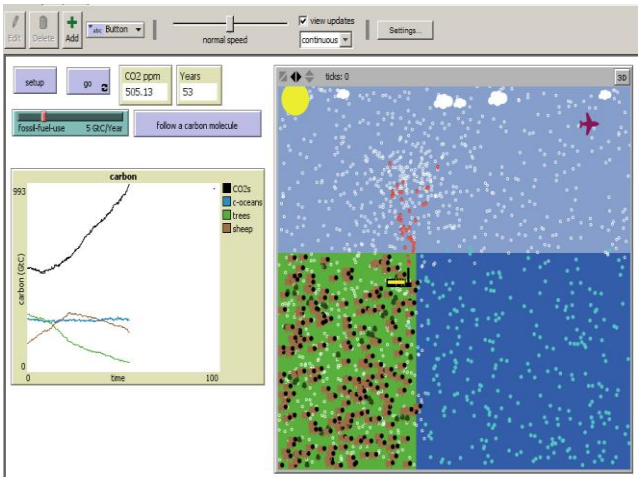

Figure 1. A carbon cycle model

2.2 Pre-processing Logs with Abstractions

Our data pre-processing included three steps. During the first two steps, we abstracted action-sequences from the event-sequences, and in the third step, we further abstracted activity-sequences from action-sequences. The recorded log files comprised process instances. A process instance is a sequence of events in which each event represents an interaction with a model (e.g., a click of "Setup" button).

In the first step, we identified five main categories of technical events: two types of basic model control; tracking of different model elements; change of modelling speed; and change of model parameters (Table 1). In our analysis we regarded dichotomous events as two discrete events (e.g., "Go" was regarded as "Start" and "Stop" of the simulation), whereas all events with continuous parameters we regarded as singular events without parameters (e.g., "Speed" was considered as change of speed without taking into account the direction of change and size). Thus, the pre-processed log files were composed from the sequences made up from seven singular technical events. In this step, process instances (sequences of events) were segmented using "Start," "Stop" and "Setup" events as the main semantic delimiters. As a result, each process instance consisted of non-overlapping segmented sequences of events. The sequences that began with "Start" and ended with "Stop" identified those events that students made during model simulation, including their implementations of planned modelling experiments and more spontaneous interactions with the model. The sequences that began with "Stop" and ended with "Start" distinguished those events that students made when the simulation was paused. We called these two types of sequences "runs" and "pauses," respectively. The execution time of each segmented sequence was computed as a time difference between the end and start of the segment. "Setup" terminated the execution of the model. Thus, semantically, "Setup" events that occurred during runs were similar the sequences of three events "Stop-Setup-Start" and was interpreted as a pause with the execution time of 0 seconds.

In the second step, we identified actions by classifying segmented event-sequences identified in the first step. Each event-sequence was considered to be "a bag of events". In order words, the order of events, except "Start" and "Stop," in each event-sequence was not considered; and all events were indexed using a binary digit. The occurrence of an event was assigned value 1, if the event happened in the sequence at least once; otherwise, the occurrence was assigned value 0 (i.e. the event did not occur). Based on this binary presentation, we clustered all segmented event-sequences into actions. Actions were represented by a string of five digits. Each event, but "Start" and "Stop," was assigned a unique position (see Table 1). The absence or presence of the event within each action was identified by assigning value "0" or "1" for the digit representing that event in the string. For example, simulation speed was represented by the second digit and fossil fuel use was represented by the third digit; thus the sequence "start-01100-stop" represented a modelling action during which a dyad changed the simulation speed and fossil fuel use while the simulation was running. Therefore, each action had a distinct semantic and represented the nature of students' interaction with the model during "runs" or "pauses."

Table 1. Types of interactions with the carbon cycle model

Buttons in the model	Category [type][1]	Technical events [position]	Examples of actions
Go [On/Off]	Model control [Dichotomous]	Start [S] Stop [E]	S-00000-E E-00000-S
Setup [nil]	Model control [Singular]	Setup [1]	S-10000-E
Follow a CO_2 molecule [On/Off]	Tracking [Dichotomous]	Follow molecule – On [4] Follow molecule – Off [5]	S-00010-E S-00001-E
Change of speed [value]	Speed change [Continuous]	Change of speed [2]	S-01000-E
Change fossil fuel use [value]	Parameter change [Continuous]	Change fossil fuel use [3]	S-00100-E

In the third step, activities were created by classifying actions. We identified activities using three features: action types, event types, and duration. Action type specified whether an activity was a "pause" or a "run". Event type specified the nature of events that occurred during each activity and included four heuristic categories: control, interaction, configuration and combined activities. *Control* activities included simple runs, pauses, and reset. They indicated those periods when students controlled the model by simply running, and pausing the simulation or resetting the simulation to the initial conditions. *Interaction* activities indicated those periods when students controlled their interaction with the model by adjusting simulation settings, such as making visible and tracking the behaviour of an individual CO_2 molecule

[1]Type - is the type of a variable in the log file corresponding different types of behaviour: *Singular events* have one discrete value (no parameter); *Dichotomous events* have two possible values (*On* or *Off*); *Continuous events* have a range of values (continuous parameters). [**]Position – identifies the position of the digit that represents each technical event in the action.

and changing simulation speed. *Configuration* activities included those periods when students changed model variables that impact the modelled phenomenon, such as the rate of CO_2 emissions. *Combined* activities included the combination of model control, interaction and configuration. Interaction and configuration activities also involved model control. To specify the time feature, we used action durations: value 1 indicated that the execution time was longer than 0 seconds; value 0 indicated that the length of an action was 0 seconds. This classification resulted in nine kinds activities (Table 2) from which we created activity logs. The three logs - the event-sequences, action-sequences, and activity-sequences – then were used as inputs to process mining.

Table 2. Classification of actions into activities

Actions[2]	Sec	Activity
S-00000-E	>0	Control_Run
S-010XX-E | S-0X01X-E | S-0X0X1-E	>0	Inter_Run
S-00100-E	>0	Conf_Run
S-011XX-E | S-0X11X-E | S-0X1X1-E	>0	Comb_Run
E-10000-S	=0	Reset_Run
E-X0000-S	>0	Control_Pause
E-X10XX-S | E-XX01X-S | E-XX0X1-S	>0	Inter_Pause
E-X0100-S	>0	Conf_Pause
E-X11XX-S | E-XX11X-S | E-XX1X1-S	>0	Comb_Pause

3. RESULTS

There were 21 event logs that, in total, included 2582 technical events. The event logs were segmented into 1520 event-sequences. Each event-sequence was transformed into "bags of events" with binary indexing. All transformed event-sequences were then clustered into action-sequences and activity-sequences.

In order to demonstrate the capability of our data preparation method, we present and compare process patterns of one dyad created using data at three different levels of abstraction - events, actions and activities. We generated three causal dependency diagrams (nets) from the pre-processed data using the Heuristic Miner algorithm with the default parameters [9, 10]. The analysis and comparison of these three causal nets shows that each of the process models depicts distinct features of students' interaction with the simulation (Figures 2-4).

The event process model shows that "start" and "stop" dominated in students' interaction with the model and each event was recorded in the log 33 times (Figure 2). "Setup" event during which students reset parameters appeared also quite often (20 times); whereas "fossil-fuel-use" and "speed" events were executed only 14 and 11 times, respectively. The model shows that "stop" event plays quite distinct role in the process pattern; and two other events - "start" and "fossil-fuel-use" - have high dependencies on "start" event (0.9). This indicates that the dyad usually took these two actions after stopping the model. "Start" and "stop" events are mutually co-dependent. It means that these students often started the simulation and then stopped it without any further interaction and/or configuration. Nevertheless, "start" and "stop" events form a part of a larger three-event loop of "stop", "fossil fuel use" and "start" with dependencies 0.9 of these three events on each other. It indicates that these students usually configured the model after stopping the simulation. "Start," "speed" and "stop" events form another loop, where "speed" event has a dependency on "start" event (0.5) and "stop" event has a dependency on "speed" (0.7). The dependencies in this event loop, however, are lower, which indicates that the change of speed did not necessary follow the start of the simulation or preceded the stop. Indeed, the extracted process net shows that "speed" also depends on "fossil fuel use," which suggests that the students sometimes changed speed just after changing "fossil fuel use" parameter, thus configured modeling and interaction parameters together. Further, "setup" event has a noticeable dependency on "stop" (0.7), suggesting that the dyad usually reset the simulation after stopping it. However, there is a loop from "setup" event to "setup" indicating that students, for some reasons, sometimes pressed "setup" button multiple times.

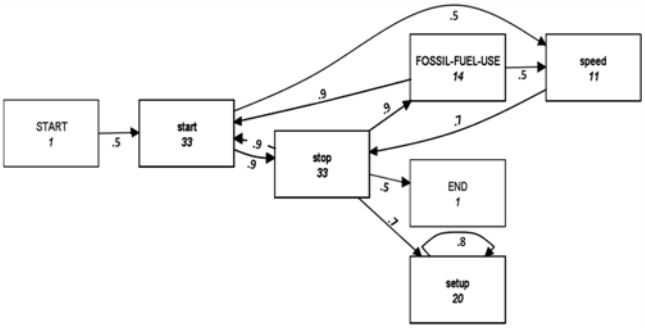

Figure 2. Process model mined from event-sequence

The action process net reveals additional characteristics of students' modeling behavior (Figure 3). The model distinguishes characteristic features of each run and each pause. Therefore, we can see what the dyad did during and in between the execution of the simulation. As Figure 3 shows, the students took only two kinds of actions when the simulation was running. Firstly, the students simply ran the simulation without any interaction and configuration (i.e. S-00000-E). This action dominated in their process model (27 times). Secondly, the students sometimes changed the simulation speed while it was running (i.e. S-01000-E), but this action appeared less frequently (8 times). In contrast, the students took a large variety of different actions between the executions of the simulation. Particularly, there are 7 types of "pause" actions. Among the six actions connected to the simple simulation run, 5 of them are co-dependent. That is, there are both out-going and in-coming dependency arrows to these five actions from the simple simulation run. This model shows that more complex interactions during the runs usually tended to follow more complex model configurations during the pauses, whereas simple interactions during the runs tended to follow more simple model reset or pause actions. It is important to note, that this tendency was not depicted in the process model that was based on the technical events (Figure 2).

[2]X represents either 1 or 0. The position in the sequences of five digits indicates that a certain event occurred during the activity: 1) setup; 2) speed; 3) fossil fuel use; 4) track a CO_2 molecule - on; 5) track a CO_2 molecule - off. Sequences that begin with "start" (S) indicate that action was performed while the model was running; whereas sequences that begin with "stop" (E) indicate actions performed while the simulation was paused.

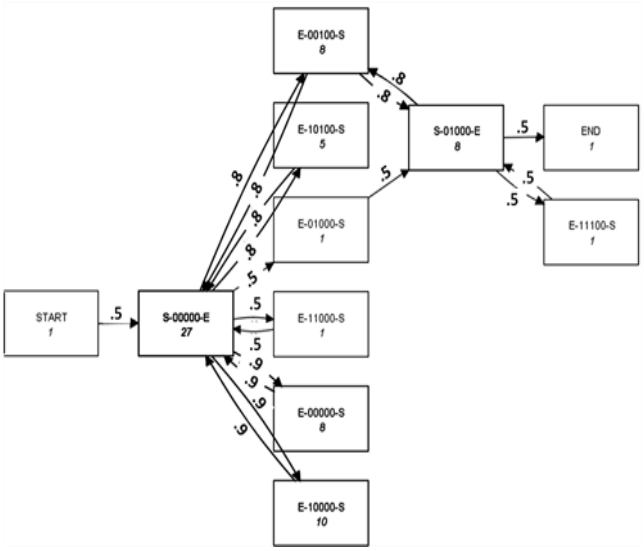

Figure 3. Process model mined from action-sequence

The activity process net provides additional insight into the students' modeling behavior (Figure 4). As we can see, students' activity pattern involves two kinds of model run activities: simple control of the model (27 activities) and interaction with the model while the simulation is running (8 activities). These activities are similar to the actions depicted in the action process model (Figure 3). However, the activity pattern depicts that students' activities during the pauses also form two dominant groups: pauses that involve simple control (16 activities) and pauses that involve configuration of the model (13 activities). The control pauses are strongly co-dependent with just one activity, which is the control of the simulation during model runs (dependency 0.9). In contrast, the pauses that involve model configuration are codependent with both types of simulation runs: simple control and interaction with the simulation while the model was running (dependencies 0.9). This pattern shows that students' active configuration during pauses was followed by a mix of passive observation and more proactive exploration during runs, whereas passive behavior during pauses was always followed by their passive behavior during runs.

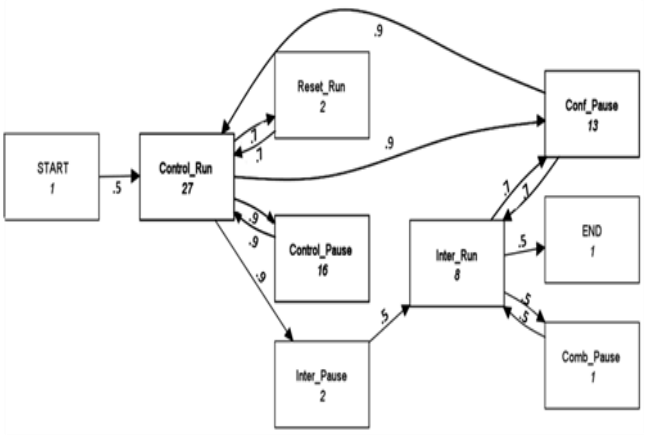

Figure 4. Process model mined from activity-sequence

4. DISCUSSION AND CONCLUSION

The technique described here presents our work in progress. In conclusion, we have created a technique to identify actions and activities based on the combination of events of students' interaction with computer models. We also illustrated that using larger semantic units and examining their sequences we could gain additional insights into how students interact with computer models.

The result presented here demonstrates the value of our technique in extracting patterns of students' interaction with computer models for individual pairs of students. Although a pattern of students' interaction can be extracted for a particular pair of students, there are different patterns for different pairs. In addition, we did not take into account students' performance and learning gain. In future, we are planning incorporate students' performance and learning gain into our study and process analysis.

5. ACKNOWLEDGMENTS

The research reported in this paper was funded by an Australia Research Council Linkage Grant No. LP100100594. We thank Paul Stokes, Nick Kelly, Kashmira Dave and the teachers and students for their invaluable assistance in this study.

6. REFERENCES

[1] Bose, R.P.J.C., and Aalst, W.M.P.V.D. 2009. Abstractions in process mining: A taxonomy of patterns. In *Proceedings of Bussiness Process Management*, 159-175.

[2] Buckley, B.C., Gobert, J.D., Horwitz, P., and O'dwyer, L.M. 2010. Looking inside the black box: Assessing model-based learning and inquiry in Biologica™, *International Journal of Learning Technology*, 5(2), 166-190.

[3] Goldstone, R.L., and Wilensky, U. 2008. Promoting transfer by grounding complex systems principles, *Journal of the Learning Sciences*, 17(4), 465-516.

[4] Kelly, N., Jacobson, M., Markauskaite, L., and Southavilay, V. 2012. Agent-based computer models for learning about climate change and process analysis techniques. In *Proceedings of 10th International Conference of the Learning Sciences, International Society of the Learning Sciences*, Sydney, Australia, 25-32.

[5] Levy, S.T., and Wilensky, U. 2010. Mining students' inquiry actions for understanding of complex systems, *Computers & Education*, 56(3), 556-573.

[6] Mcelhaney, K.W., and Linn, M.C. 2011. Investigations of a complex, realistic task: Intentional, unsystematic, and exhaustive experimenters, *Research in Science Teaching*, 48(7), 745-770.

[7] Pedro, M. S., Baker, R.S.J., Gobert, J.D., Montalvo, O., and Nakama, A. 2013. Leveraging machine-Learned detectors of systematic inquiry behavior to estimate and predict transfer of inquiry skill, user model. *User-Adapt. Interact.*, 23(1), 1-39.

[8] Thompson, K., and Reimann, P. 2010. Patterns of use of an agent-based model and a system dynamics model: The application of patterns of use and the impacts on learning outcomes, *Computers & Education*, 54(2), 392-403.

[9] Weijters, A.J.M.M., and Ribeiro, J.T.S. 2011. Flexible heuristics miner (FHM). In *CIDM'11*, Eindhoven University of Technology, Eindhoven, 310-317.

[10] Weijters, A.J.M.M., van der Aalst, W.M.P., and Medeiros, A.K.A.D. 2006. Process mining with the heuristics miner algorithm, *Technology*, 166, 1-3