TOWARDS AUTOMATIC BEHAVIOR ANALYSIS OF LEARNERS IN A TECHNOLOGY-ENHANCED LEARNING ENVIRONMENT

Noury Khayat, Michael Mock and Jörg Kindermann
Fraunhofer Institut IAIS, Schloß Birlinghoven, Sankt Augustin, Germany

Keywords: Agents, Analysis, Behaviour monitoring, Technology-Enhanced Learning Environment, Workflow.

Abstract: In the context of the European SCY project, a collaborative, learner-centric TEL environment is described. Reference workflows and workflow executions are analyzed to extract meaningful behavioural attributes automatically. The extracted patterns provide insights into learner behaviour.

1 INTRODUCTION

Technology-Enhanced Learning (TEL) Environments are systems designed to support learning and teaching, based on the support of new emerging technological achievements. An approach to achieve such a TEL Environment is being done in the SCY Project In SCY-Lab, the TEL environment produced by the project, learners work on a given "mission", like an experiment, a design task, etc., to reach the intended goals in the learning process. A mission can be any complex task or assignment with an interdisciplinary scientific background, guided by a general question (e.g.”How can we build a CO2-neutral house?”, or “Development of a healthy menu for the school canteen”). The fulfilment of a mission requires a combination of individual and collaborative contributions from the learners, using a provided set of tools, services and resources.

Teachers expect specific results from learners during and after mission execution in the form of a text, an image, a design, a simulation model of a physical system, etc. Those missions, represented as workflows, are executed based on provided rules and semantics. We call this a workflow model. Each execution of the model is called a workflow execution. We can summarize the building blocks of a workflow model as follows: (1) Scenarios: Each learning mission has one scenario object as an encapsulated block, which represents all the tasks to be executed, (2) Learning Activity The primitive unit which represents the requested task to be executed, (3) Learning Activity Space (LAS) (Ney et al., 2009): Grouping objects, which contain other objects on lower levels, like the learning activities. (4) Emerging Learning Objects (ELOs): represent the pedagogically relevant products of learner performance during the learning process. (5) Tools: components of the learning system used in creating, manipulating or storing ELOs. (6) Action logs: all interactions of the learners with the system when executing a workflow. (7) Scaffolds: feedbacks or hints provided to the learners.

SCY contains a high level of interactivity: Interaction between learners is a key point in improving their learning performance. Interaction of learners with SCY technology is necessary to produce the ELOs. In SCY, pedagogical agents (i.e. autonomously active components of the learning system) are used to analyze the interaction of the learners with the technology. The agents send scaffold to the learners automatically.The interaction of the teacher with learners is necessary for supervision. The interaction of teachers with the SCY technology also is essential, since the teachers have the responsibility of shaping the pedagogical missions and monitoring the learners' executions.

Along those interactions, the learners leave specific traces as action logs of the executed workflows. Valuable insight about the execution is therefore hidden in the actions logs. Manual inspection is not possible because of the huge amount of data. Automatic analysis is needed to extract informations, which reveal behavioral aspects, the learner has shown during the execution of the mission. The teachers and pedagogical experts

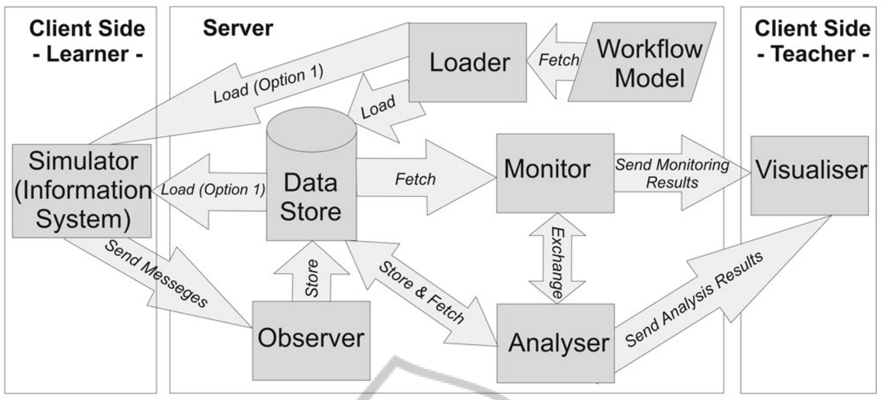

Figure 1: System architecture.

will thus be able to obtain answers to pedagogical questions.

This paper features methods and techniques to extract that information automatically from the action logs. Meaningful patterns, potential outliers and behavioral aspects are results of the extraction process.

2 RELATED WORK

2.1 Learning Environments

The DORIS pedagogical agent for intelligent tutoring system is an approach to follow learners' interaction with the tutoring system, and guide them during the learning process (dos Santos et al., 2002). Those agents perceive related information about the learners' interaction with the system, and make an appropriate response based on that information. DORIS agents take into consideration specific perspectives during the monitoring process. Those perspectives are restricted to starting and finishing time of the interaction incidents between learners and the system; pages visited by them and the consumed duration in each of those pages.

In SCY, the learning missions are represented as workflows, to be executed by learners in computer systems. That allows providing a more specific plan for the learners to follow (de Jong et al. 2010).

2.2 Behaviour Modelling

Behaviour modelling is a semantic abstraction of some observed actions to fit a model, which caused the generation of those actions. In (Bollen, Giemza, Hoppe 2008) an architectural framework for distributed collaborative learning environments with agent support is described which features rule-based state and action pattern analysis. (Akhras, John, 2002) discusses intelligent tutoring systems from a constructivist perspective, which includes a domain model, a teaching model, and a learner model, therefore exceeding the SCY approach, which does not include full user-modelling.

In (Zhou, Evens, 1999), manual experiments have been done in the context of the CIRCSIM-Tutor intelligent tutoring system, to define what a learner model is, and how to divide the learner model into components.

Another approach has been presented in (Fernandez et al., 2009), where a methodology has been developed as an alternative to discover human behaviour patterns using automatic learning methods. The approach reveals the execution of the workflows which have been executed by converting gathered sensor data to the form of a workflow.

3 SYSTEM ARCHITECTURE

An appropriate distributed client-server architecture has been designed for the SCY system to make it accesable from different places. The client, SCY-Lab, is to be used by the learner or the teacher. The back end is the SCY server which provides the services for the learner and the teacher, and is supports the analysis processes. Accordingly, we have designed our system as a distributed system, which contains three functional parts: an information system at the client side, including a graphical user interface (GUI) for the teacher and pedagogical experts, and the monitoring and analysis system at the server side.

As there was no complete running version of the SCY system early in the project life-cycle, we had to simulate the learner behavior and his execution of the mission. The system is separated into several modules, which communicate with each other, to serve in monitoring a specific scenario. Figure 1 shows an overview of the modules of the system, distributed on the appropriate architecture sides.

A standard working scenario of the system starts at the loader, which loads the SCY workflow model into the permanent data store as a reference, and also into the simulator to be executed by the virtual learners.

Once the model has been loaded into the data store, the simulator can, but does not have to, use the stored model directly without the need of reloading it.. In the simulator, the simulated learners start executing the workflow model, by using a workflow engine, and firing meaningful events that describe the current state of the execution. A generic observer at the server side catches those events, filter them based on configurable settings and stores them in a dedicated permanent data store. This allows the monitor and the analyzer to access execution data. The results are directed to the visualiser.

In this way, we can easily integrate our monitoring and analysis system into the SCY System, relacing the simulation system by real learners. The monitoring and analysis tools and utilities would be represented as one service in the main SCY server as a kind of Service Oriented Architecture (SOA). More details about the SCY-Lab architecture are available in (de Jong et al. 2010).

The Workflow Management Coalition (WfMC) has defined a workflow management system (WfMS) as "a system that defines, creates and manages the execution of workflows". Therefore, our system can be considered as a workflow management system.

4 WORKFLOW ENRICHMENT MODEL

Workflow models are provided to the system, executed by the workflow engine in the simulator, events are fired describing the states of the execution, and analysis and monitoring is done on the workflow execution. The common factor in all those scenarios is that everything is done relatively to the workflow model. Therefore the processing results can be seen as extensions to the workflow model. We call this "workflow model enrichment". Substantially, the enrichment is done by interpreting the outcomes of the executions relatively to the workflow model, breaking them into smaller units, transforming them and getting meaningful information out of low-level data. These functionalities are implemented in the analyzer module. First we describe the processing of simple events. Then, complex events as patterns of simple events or workflow objects, are investigated. Finally, our understanding of the processing results and possible applications based on different perspectives, are presented, resulting in the "Behavior Modeling".

4.1 Simple Events Processing

Different states can be assigned to the different workflow objects depending on their semantics during the execution phase or afterwards. That can happen either directly by an action of the learner or as computation of a specific behavior issue.

States are triggered by events. The events can be categorized by: (1) the types of workflow objects, on which the events occurred, like: Activities, ELOs, etc.; (2) the perspective of the access operations to the workflow objects, like: reading, writing an object, etc. (3) depending on the conditionality of firing the events. There are unconditional types of events, which will be fired independently of any parameters, and conditional types of events, which will be fired only if specific criteria have been met, like a parameter value exceeding a threshold, etc.

4.2 Complex Events Processing

The cloud of simple events, makes the task more diffcult to get a meaningful insight into the executions of the workflows. Therefore we add a higher level by combining various simple events to form a complex event. Complex events can also be organized in hierarchal structures, where a complex event can contain other complex events. For example, firing a complex event, related to the whole mission, depends on the fired complex events of the contained executable workflow objects. The complex events are analysis units, which are used in this work in two contexts: event patterns and workflow patterns.

4.2.1 Event Patterns

Event patterns are sequences of simple events. An example is an essential event pattern, which expresses the visit concept of an executable workflow object, and includes the simple events of entering and leaving that object.

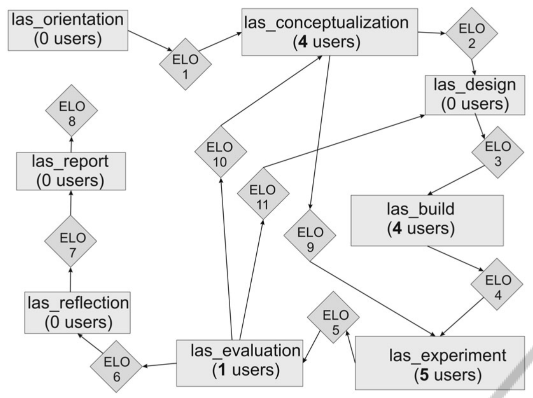

Figure 2: A sample output of the workflow visualiser.

Event patterns are also helpful in firing some simple events, which would not be possible to detect without their support. For example, the event of exceeding the expected execution time for an executable workflow object could be fired by comparing the duration of the visit event pattern with the reference value of the expected time of that object.

Thus, sequences of specific event patterns could result in producing a higher level pattern, which reflects the structure of the executed workflow objects. We call the higher level patterns workflow patterns.

4.2.2 Workflow Patterns

Important approaches in the field of workflow patterns are discussed in(Russell, et al. 2006) . For SCY, specific workflow patterns have been developed.

An example of workflow patterns is the loop, a group of executable workflow objects, where the execution is done in a sequential way, and can be repeated. In SCY, we recognize two types of loops, being inspired by (Russell, et al. 2006) :

- Safe loop: This kind of loop has only one entry and exit point. Once an iteration of a loop has started, it is not possible to be interrupted by leaving the loop from some exit point other than entry point. This loop corresponds a structured loop in (Russell, et al. 2006) .

- Unsafe loop: This kind of loop has more than one exit point, independently of the number of entry points. Unlike in a safe loop, it is possible to leave an iteration, without completing it to the same point where it has started.

Workflow patterns cab be seen in SCY on two levels: (1) as workflow models, where the patterns are extracted automatically from workflow models, and stored as references in the data store for later use. (2) as workflow executions, where the patterns are detected during the execution through the events, and referred to as instances of the workflow patterns on the workflow model level.

Loops are extracted from workflow models by traversing the workflow, and building a tree of the traversed workflow in a specific way, which reveals the existence loops. It is an optimized version of the algorithm of (Havlak, 1997), where the Havlak's algorithm does not remember the workflow objects already traversed. The set of the workflow patterns which the learner has followed to reach the goal can be used to extract higher level behavioral aspects of the execution.

4.2.3 Behaviour Modelling

The learner executes the workflow and creates his own register of execution outcomes, which define the level of quality according to constraints, rules and hints. Modeling the behavior can be seen from several perspectives.

(1) The time perspective: the workflow model should contain all the information required by the learners, including the expected time to be consumed. It is possible to infer if the behavior of the learner is accordant to the providedconstraints.

(2) The routing perspective is based on the paths in the workflow, or the set of the workflow patterns, which the learner has followed to reach the goal. The workflow model should be provided with a specified optimal path, which could be considered as a reference for the learners.

Now we present the process of extracting the attributes which represent the behavior of the learner, the behavioral attributes. Our approach depends on statistical computations of the accumulated information and analysis results of the workflow execution. From the time perspective, we define two new concepts, the “time out” and “time rest”, as follows:

(1) *Time Out* is the time the learner has consumed in an executable workflow object, after exceeding the expected consumption time. It is always accompanied with an “exceeding time” event.

(2) *Time Rest* is the time which the learner is still allowed to consume in an executable workflow object, at the moment of finishing that object. It is always accompanied with a “finishing in time” event.

Those concepts are applied to both the single visits and all the visits; and the statistical computations are calculated on both concepts separately.

From the routing perspective, we depend on aggregation meausrements of the executed workflow objects, relations and patterns. The calculation those routing measurements is based on two secondary perspectives:

(1) Correctness is considered as a precision-oriented concept. We distinguish between executed workflow patterns which have been executed according to the systematic or optimal behavior and others. The former are consided to have a feature of "correctness". Note that we assume “optimal” workflow executions to be denoted explicitly by the pedagogical expert when designing the SCY learning workflow.

(2) Counting occurences of patterns: We differentiate between the fact whether a specific workflow pattern has been executed and the number of executions. The former is called a “class”, the latter an “instance” (in analogy to object-oriented modeling). For example, the aggregation of a specific workflow pattern which has been executed N times, would generate the value 1 for the "class" measure and the value N for the "instance" measure. The extracted behavior attributes are abstracted into concrete behavioral concepts:

- *Correctness* is an outcome of the behavior modeling, which is obtained from the routing behavioral attributes as seen from the "correctness" perspective. It specifies how the execution corresponds to an optimal behavior. The different attributes of the correctness vector have different meanings depending on the patterns contributing to the attribute: loops, sequences, etc.
- *Repeatness* is obtained from the routing behavioral attributes as seen from the "abstraction" perspective. It specifies how the workflow objects are repeated in an execution.
- The *time-related* measure represents the temporal behavior of the execution as obtained by fetching the time attributes from the extracted attributes.
- *Loopness* expresses how “loopy” the execution is: the number of executed loop classes, loop instances and correct. It is obtained by fetching the loop-related attributes.
- *Sequentiality* expresses how sequential the execution is. It is of minor importance than loopness, since the semantics of sequences are not as critical as loops; i.e. an infinite loop would lead to failure, and that is not true for the sequences.
- *Conjunctionality* expresses the behavior of executing conjunctions as splits. This measure gives an impression on the the transitivity of workflow execution; i.e. how many redirections the learner has taken during the execution.
- *Disjunctionality* expresses the behavior of executing conjunctions as joins. This measure has an opposite meaning of Conjunctionality.

Other combinations of the extracted attributes could be evaluated, based on the required semantics and purposes and by applying the appropriate workflow patterns.

Table 1 is a part of the data table of low level behavioral attributes. The learner U2 has not finished the mission in time. It appears that he was too "loopy" during the execution (number of executed loop classes is 2; number of the correct loop classes is 1). Similarly, the number of correct loop instances is 2, whereas the number of executed loop instances is 4. Similar to U2 are U4 and U5. The difference is that U5 has executed more wrong loop instances than U2 and U4, but that has not led to worse than exceeding expected time.

Table 1: A low level behavioural attributes data table.

	# of loops		# loops correct		finished
user	class	instance	class	instance	in time
U1	2	2	2	2	yes
U2	2	4	1	2	no
U3	0	0	0	0	yes
U4	2	4	1	2	no
U5	2	6	1	2	no

An opposite example is U1. He has finished the mission in time, executed loops from 2 loop classes, and both are included in the optimal behavior. Similar to U1 is U3. The is that U3 has not executed loops at all, which has apparently led to finishing also in time.

The discussed behavioral attributes show potential explanations of the mission outcome. Attributes from other workflow patterns have also been investigated and implemented. The low-level attributes can already a first insight, but they primarily serve as input to further data-mining analysis.

5 MONITORING AND ANALYSIS TOOLS

Our monitoring and analysis methods and techniques should be usable by the teachers and pedagogical experts. Therefore a GUI is needed which allows the teachers and pedagogical experts to interact with the monitoring and analysis system, in addition to the visualization techniques that deliver the extracted knowledge insights. The visualization tools are:

(1) The *Logger* exposes the details of the current running processes and operations.

(2) The *Workflow Visualiser* revisualizes the workflow model with up-to-date execution data.This tool is useful to feedback the teachers with an online overview of the current execution details in a visual form (Figure 2). The number of current learners in each workflow object is shown. The size of the workflow object is relative to the number of executing learners, to make the monitoring process easier for the teacher.

(3) The *User Follower* is similar to the workflow visualiser but concentrates on delivering individualinformation about the execution for specific learners, like statistical information and behavioral attributes.

(4) The *History Visualiser* allows the teacher to track the history of execution for each learner visually on a time axis.

(5) The *Query Builder* allows to create a specific high-level query on the stored data in the data store. That is done by a GUI that is configurable to map a parametriezed query to a low-level query that can be executed directly at the data store side.

6 CONCLUSIONS

In this work, we have presented an approach to monitor and analyse the execution of missions, representd as workflows, in a TEL enviroenment and an approach towards an automatic analysis of the learners' behaviors.

The developed system is able to load SCY workflow models, to provide generic representations of the semantics of those models. A generic data store has been designed and developed to store those representations. That property of generality enables successful storing, monitoring and analysis of workflow models of other environments than SCY,
in case of extending our system for other workflow environments. As there was no running version of SCY system at the moment of doing this work, a simulator has been designed and developed to simulate the execution of loaded and stored workflows, to provide the required data for the monitoring and analysis processes. Patterns in the workflows models and executions have been defined, extracted and used for monitoring and analysis processes. Techniques to extract behavioral attributes have been defined as an approach towards modeling the behavior of a workflow executor depending on time and routing perspectives at a semantic level.

ACKNOWLEDGEMENTS

The fulfilment of this work has partially been supported by the SCY project, www.scy-net.eu , a EU- funded IP project, grant agreement number 212814, FP7.

REFERENCES

Akhrasi, F., N., Self, J., A., 2002. Beyond intelligent tutoring systems: Situations, interactions, processes and affordances. In: *Instructional Science* 30: 1–30

Bollen,L., Giemza, A., Hoppe, U.,2008. Flexible Analysis of User Actions in Heterogeneous Distributed Learning Environments. In: *Proceedings of the European Conference on Technology Enhanced Learning.*, Maastricht

de Jong, T., et al. 2010. Learning by creating and exchanging objects - the SCY experience. In: *British Journal of Educational Technology*, 41, 909-921.

dos Santos, C., Frozza, R., Dhamer, A., Gaspary, L., 2002. Doris| pedagogical agent in intelligent tutoring systems. In: *Intelligent Tutoring Systems*, 91-104.

Fernandez, C., Lazaro, J., Benedi, J., 2009. Workflow mining application to ambient intelligence behaviour modelling. In: *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments.*, Springer

Havlak, P., Nesting of reducible and irreducible loops. 1997, in: *ACM Transactions on Programming Languages and Systems*, Vol. 19.4, 557-567.

Lejeune, A., Ney M., Weinberger A., Pedaste M., Bollen L., Hovardas T., Hoppe U., de Jong T., 2009. Learning activity spaces: Towards flexibility in learning design. In: *Ninth IEEE international conference on advanced learning technologies.*

Russell, N., ter Hofstede, A., H., M., van der Aalst, W., M., P., Mulyar, N., 2006. Workflow control flow patterns: A revised view. In: *BPM Center Report BPM0622 BPMcenter org*

Zhou, Y., Evens, M., W., 1999. A practical student model in an intelligent tutoring system. In: *Proceedings of the 11th IEEE international conference on tools with artificial intelligence*, 13-18.