Using Capture & Replay for Semi-automatic Assessment

Daniel Herding, Ulrik Schroeder

Computer-Supported Learning Research Group

RWTH Aachen University

Abstract

Semi-automatic assessment is a concept that attempts to unify the advantages of Computer-Aided Assessment and tutorial feedback. In the sense of formative assessment, the student should be able to demand feedback not only on her final solution, but also during her entire solution process. The Jacareto capture & replay tool was used to record student interaction with a learning application, with the goal of making the learner's approach comprehensible to the tutor, and thereby making elaborate feedback possible. This contribution describes challenges that had to be overcome to realize a semi-automatic assessment scenario with Jacareto.

One major advantage of Computer-Aided Assessment (CAA) is that students can get feedback directly from the learning application, without having to wait for their tutor to review their solutions. However, a manual review is still necessary for complex, open-ended tasks that cannot be assessed fully automatically. This includes tasks like proving a mathematical theorem or programming a graphical user interface (GUI).

The SAiL-M project[1] aims for a synthesis of automatic and manual assessment. The idea of so-called semi-automatic assessment is that the computer delivers feedback

[1] SAiL-M project: http://sail-m.de/english. Funded by the German Federal Ministry of Education and Research.

on those aspects of the solution that it can assess, and delegates any further questions to the tutor. This eases development of learning applications, as they only have to detect standard solutions and common mistakes. Any unconventional solutions or unexpected mistakes are handled by a human tutor. Ideally, the tutor can fully concentrate on these cases, as he does not need to support the other students (Bescherer et al., 2011).

One learning application that supports semi-automatic feedback is SetSails! (Herding et al., 2010), which deals with set algebra. The student has to perform a sequence of algebraic transformations to prove the equivalence of two terms. At any time, he can demand feedback on her intermediate solution. While the application can evaluate whether a step follows one of the built-in rules (e.g. commutative law), it cannot assess transformations with learner-defined rules. In such a case, the student is asked to send her (intermediate) solution to her tutor for revision (see figure 1).

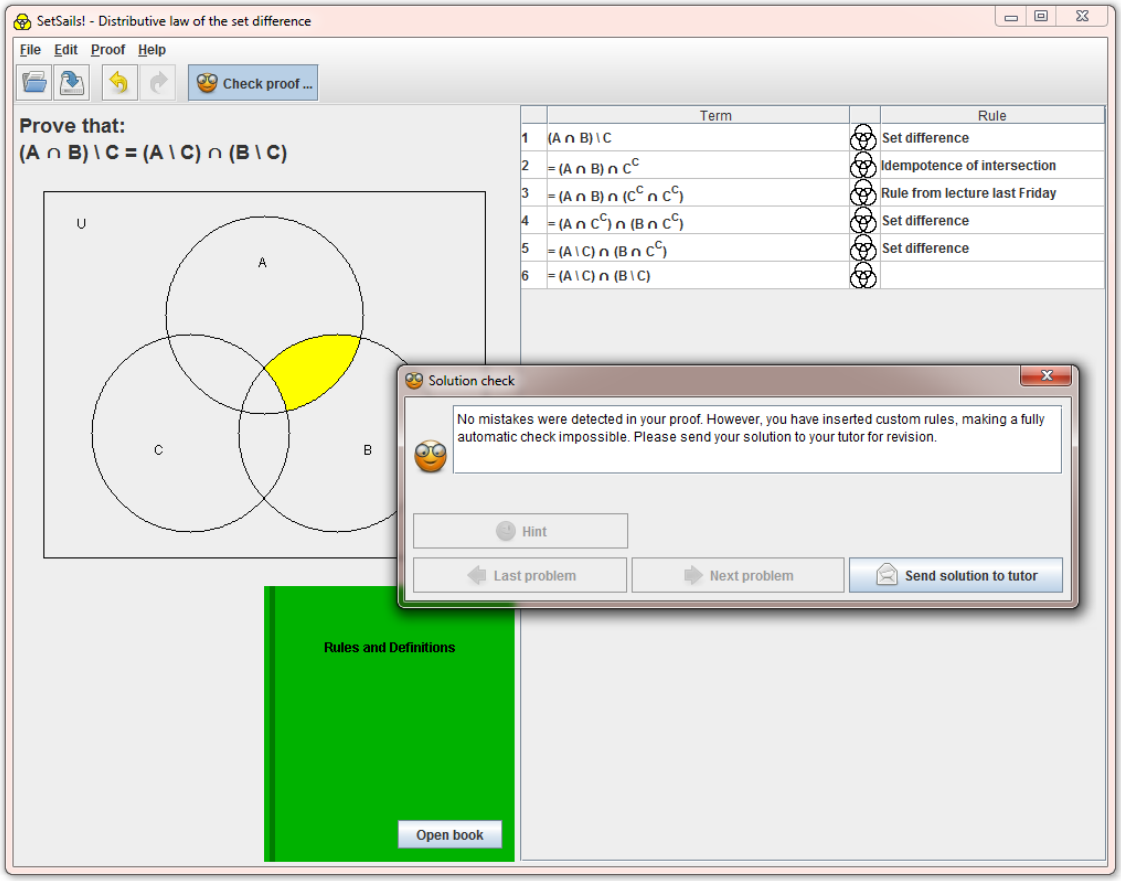

Figure 1. The feedback dialogue in the SetSails! application. A learner-defined rule was used for the transformation in line 3.

Even though a tutor may be able to detect any mistakes by just looking at the intermediate solution, it may be hard to tell why these mistakes have been made. This makes it difficult to give elaborate feedback. To comprehend the cause of a mistake, one needs to look into the steps that led to it. "One of the benefits of CAA is the opportunity to record student interactions and analyse these to provide a richer understanding of learning" (Conole & Warburton, 2010). Evaluating a single student's record enables the tutor to give her elaborate feedback. Evaluating several students'

records even makes it possible to find common misconceptions and to adapt teaching strategies accordingly.

Related Work

Mason and Bacsich (1997) describe a scenario in which tutorial support is offered via computer conferencing. Students can join a conference to query their tutors or to discuss assignments. Although this was beneficial especially for distance students, hosting (and encouraging participation in) regular conferences was much too time-consuming for tutors.

As an alternative which does not rely on synchronous communication, students can use screen recording software to capture their solution processes, and send the recorded videos to their tutors. Abdel Nabi and Rogers (2009) conducted a study in which students recorded annotated videos of their assignment solutions and uploaded them to a Learning Management System (LMS). Tutors could watch these videos to assess the students' "procedural competency" and "conceptual understanding", rather than just the correctness of their outcome.

Nevertheless, one has to note that, for exercises that take several minutes to solve, such videos become very large files. This means that it takes a lot of time to encode and to upload them. In the study of Abdel Nabi and Rogers, students had three weeks to work on the exercise and finally had to record only a single video. If one wanted to offer semi-automatic feedback during the entire solution process, the encoding and uploading time would become major drawbacks. Furthermore, video files do not carry any semantic information, which means that the only way to assess them is to take one's time and watch them.

FORMID-Observer is a logging mechanism that does not allow the tutor to see the student's view. Instead, it records "semantic events" that give an insight of "learner achievements and difficulties" (Guéraud et al., 2009). Semantic events allow for analysing processes automatically, but it may be hard for a human tutor to read and understand log files if automatic analysis mechanisms fail. What is needed is a tool which combines video and log file analysis in order to benefit from human as well as computer-based intelligent feedback.

The Jacareto Framework

The acronym "Jacareto" stands for "Java Capture and Replay Tool". It is a Java framework that makes it possible to capture a user's interaction with a target application, and to replay it later. For instance, the student can use Jacareto to capture her interaction with a learning programme. The teacher can later replay this interaction to retrace the solution process (Schroeder & Spannagel, 2006).

The capturing engine of Jacareto is based on the event principle of the Java platform. For each user interaction (e.g. mouse motion, key stroke, button click), Java generates an event that the learning application processes. Jacareto intercepts each event and creates a corresponding so-called recordable. Recordables are automatically grouped and added to a tree structure, which can be stored in an XML file. After recording, the file can be transferred to the tutor's computer.

In order to replay the record file, the tutor opens it in CleverPHL, the graphical front end of Jacareto. Each recordable is converted back to an event, which is then processed by the learning application. In effect, the tutor sees the mouse cursor

move automatically, recreating all the student's actions. This information enables the teacher to comment not only on the current solution state, but on the entire solution process.

Besides the possibility to replay the record, the tutor may also filter certain recordables and export them for statistical analysis. For example, he may be interested in which mistakes are made most frequently.

Using Jacareto for Semi-automatic Assessment

Schroeder and Spannagel (2006) state that "[in] principle, you can take any interactive software written in Java and use it together with CleverPHL". However, they restrict their own statement in the same paper. "Although many applications written in Java can be used together with CleverPHL, there are some problems which restrict the practicability of CleverPHL" (ibid.). This section highlights some challenges in implementing a Jacareto-based mechanism for semi-automatic feedback inside the SetSails! learning application.

Deploying Jacareto

In a survey conducted at Ludwigsburg University of Education, 10 out of 88 participating mathematics teacher students (11 %) stated that they could not use SetSails! because they had problems installing it on their private computers. The installation procedure, as described on the exercise sheet, consisted of unpacking a ZIP file and double-clicking a JAR file. This shows that even small technical challenges can lead to a significant loss in participation numbers.

If the students had been asked to install not only SetSails!, but also Jacareto, the drop-out would certainly have been even higher. After installing Jacareto and starting CleverPHL, one has to open a new session and create a so-called starter. The starter gives Jacareto the information needed to execute SetSails!, e.g. the installation path of SetSails! (Spannagel 2007, p. 173). After manually starting the capturing process, one can use SetSails! to solve the exercise while CleverPHL records in the background. Finally, one has to stop the recording, save the session, and send it to the tutor. All these actions add extraneous cognitive load (Sweller & Chandler 1991), distracting the student from the learning tool itself. Furthermore, the effort discourages her from handing in recordings of intermediate solutions in case questions arise during the solution process.

For these reasons, a different way of deploying Jacareto was required. As an alternative to the CleverPHL user interface from which the learning application can be started, Jacareto can now be embedded into the learning application as a library. SetSails! starts recording automatically once an exercise is opened, and stops when it is closed. The Jacareto integration is completely transparent to the user, leaving cognitive capacities free for the learning task. Besides, no Jacareto installation is required anymore because it is shipped along with the SetSails! application.

Combining sessions

Most previous research projects which relied on Jacareto used it to capture continuous sequences of learning. For example, Spannagel (2007) recorded pupils' interactions in a number line game. The task – marking several integers on a number line – can be solved in a few seconds. Therefore, there is no reason for a pupil to interrupt this task and resume it later.

In contrast, the idea of semi-automatic feedback requires exactly that. A student who is stuck in her solution process and requests feedback cannot expect an immediate response from her tutor. Thus, she needs a feature to save the current state of the exercise and to reopen it after receiving feedback. The tutor needs a record of the entire solution process, not only of the last session.

In SetSails!, this was accomplished by storing the record file inside the exercise file. When the exercise file is reopened, the record is not overwritten; instead, new recordables are appended to it. In effect, all sessions in which the student worked on the exercise are combined into a single record file, which the tutor can conveniently replay.

Transmitting records

The ability to easily transfer Jacareto records from one computer to another is crucial for semi-automatic assessment, as the learner should be able to send the record to the tutor without effort. Transmitting it over the Internet is an obvious choice.

CleverPHL used to save each session in a directory that contained several XML files. In the course of the SAiL-M project, the format was changed so that each session is now saved as a ZIP file which holds XML files. This format has two advantages. Firstly, when using typical network protocols, sending a single file is much simpler than sending an entire directory. Secondly, the ZIP compression reduces the overhead that was a problem with uncompressed XML files (Spannagel 2007, p. 248). Notably, compressed Jacareto records are significantly smaller than the screen videos discussed in the "Related Work" section. For example, a five-minute Jacareto record has a file size of about 150 KB, thus can be easily transferred over the Internet.

While choosing a transfer protocol, one envisaged approach was to upload Jacareto records to the LMS used in the respective course. Unfortunately, the involved universities use different LMSs. Until we have finished developing a common interface for these, we are sending records as email attachments via SMTP.

SetSails!, like other SAiL-M learning tools, offers a feedback dialogue window that the student can open at any time to check her intermediate solution. This dialogue displays mistakes that have been detected by the automatic tests of SetSails!, and (when possible) offers hints on how to fix them. This feedback dialogue is part of the Feedback-M framework (Herding et al. 2010). The dialogue also includes a button labelled "Ask tutor a question" which makes it possible to send an email directly from the learning application. This button is supposed to make it easier for students to contact their tutors, and should encourage them to do so.

When sending an email, the student has the option to attach the current state of her exercise. As described earlier, the exercise file contains a Jacareto record. Thus, the tutor is enabled to watch the replay whenever a student asks for help, and thereby comment on the student's approach in a reply email. This process is shown in figure 2, a concretization of the Intelligent Assessment Model described by Bescherer et al. (2011).

The tutor can also collect several records and analyse them together, e.g. to look for common misconceptions. Unfortunately, such research would only include solutions of those students who contacted their tutor. It would be technically possible to quietly send an email whenever a student closes SetSails!, but this raises serious privacy concerns. Until a model is found that conforms to data privacy regulations, emails will only be sent on student request.

The changes to the Jacareto framework that were described so far are primarily learner-centred, as they remove the burden of controlling the recording process. For replay, the tutor manually extracts the record file from the received exercise file and opens it in CleverPHL.

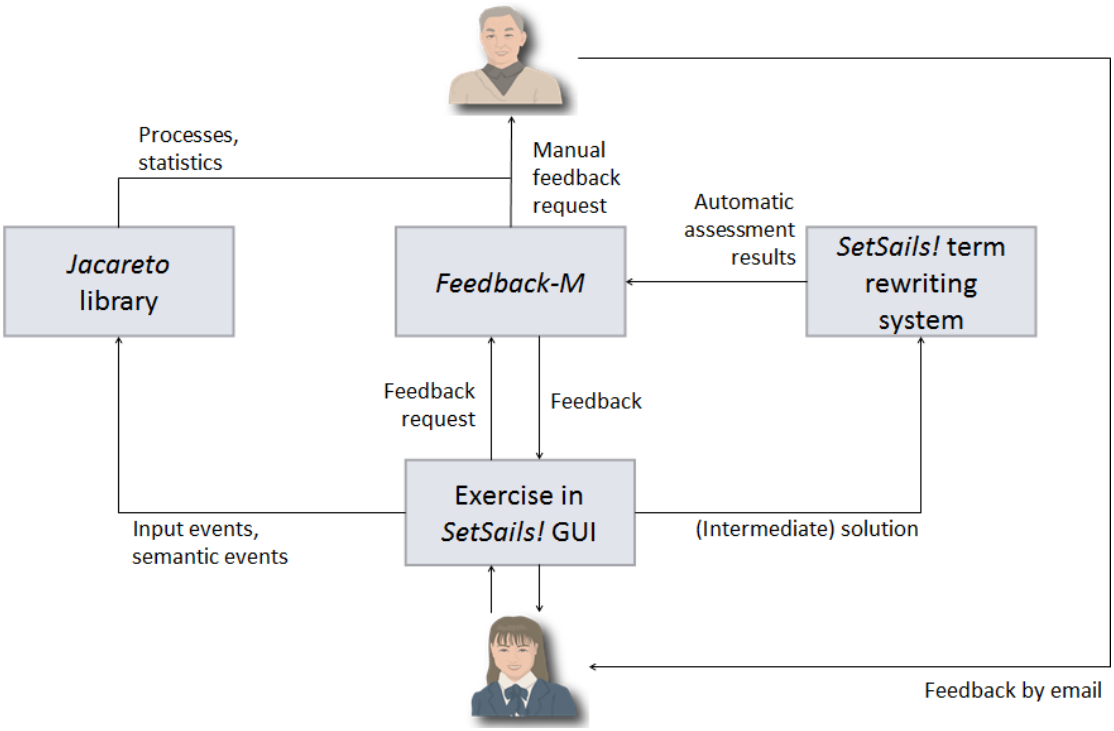

Figure 2. Feedback processes in the SetSails! application.

Analysing semantics of records

Even though watching a replay might help the tutor to comprehend a student's difficulties, one has to keep in mind that it takes a lot of time to watch the replays in full. The tutor may use the fast-forward feature of CleverPHL to jump to a certain situation, but it is not easy to find the relevant point in time.

Instead of watching the replay, the tutor could just look at the list of recordables. However, it is difficult to make any sense of events such as mouse clicks or key strokes. This is because the events that Java generates convey no semantic information. In order to augment the Jacareto replay with human-readable annotations, one needs to modify the source code of the learning application. Whenever something happens that may be relevant for a tutor, the application must push a so-called semantic event into the record (Spannagel & Kortenkamp 2009).

Before, adding a new semantic event type was very laborious. A software developer had to create an event class, a recordable class, and a converter that translated the recordable to XML and back again. In order to support quantitative analyses, she also had to programme a filter that made it possible to export the semantic events to a statistics tool. In addition to that, she had to create a configuration file that defined how these classes interrelate, and pack the configuration and class files into a module that had to be deployed along with the learning application.

These requirements made it very hard to add a sufficient number of semantic event types to a learning application. This task was simplified with the introduction of generic semantic events to the Jacareto framework. Since then, a software developer no longer needs to create any classes, configuration files, or modules to integrate semantic events. Instead, she only has to add some lines of code at the position where the event occurs. Listing 1 exemplifies this concept:

```
1  SemanticEvent event = new SemanticEvent(this,
2          "ProblemReportedEvent",
3          "The feedback dialogue reported a problem.",
4          "Problem", problemText,
5          "Hint", hintText);
6
7  TargetApplicationEventObjectQueue.pushEvent(event);
```

Listing 1. Adding a semantic event to the record.

This generates an event which tells that the student has read a problem report in the feedback dialogue. The first lines create an event of type "ProblemReportedEvent" with a human-readable description. Lines 4 and 5 define the attributes of this event, making sure that the tutor can later read the exact problem report and hint text that has been displayed to the student. In line 7, the event is pushed onto a queue which will forward it to the record. For other semantic event types, the developer may choose to use a different number of attributes.

Excluding parts of the record

It is generally desirable for a replay to resemble the original interactions as close as possible. However, there are some exceptions to this rule. For instance, the SetSails! application offers a print feature. The fact that students use it might be interesting for the tutor (for this reason, a semantic event is generated whenever the student prints her solution). However, the tutor would certainly not want his own printer to to waste ink and paper each time he watches a student use the print function. This is why a mechanism was needed to exclude certain parts of the record from the replay.

This was accomplished by adding two special types of events, named *InterruptReplayEvent* and *ContinueReplayEvent*. The learning application must generate an interrupt event just before the print dialogue is opened, and a corresponding continue event once it is closed. Furthermore, the developer has to modify the learning application so that the "print" menu item has no effect when replaying. The Jacareto replay omits everything that happened between the interrupt and continue events (i.e. the interaction with the print dialogue).

Similar changes had to be made for other parts of SetSails! that should be included from replay, such as the "Save file as…" dialogue. One other important action that should be omitted is the one which causes an email to be sent to the tutor. Otherwise, the tutor would keep sending himself an email each time he watched a replay.

Even though this approach works reasonably well for SetSails!, it is desirable to find a more elegant way of skipping parts of the record. For example, it might be possible to augment Jacareto so that it automatically detects that a newly opened window is a print dialogue and skips all interaction with it.

Recording unconventional GUIs

Jacareto was designed to record and replay interaction with any GUI that uses the AWT or Swing APIs (i.e. the two GUI toolkits included in the standard Java distribution). Nevertheless, not all aspects have been thoroughly tested yet. Replay usually works fine for simple GUIs, such as the number line game mentioned earlier, which only consists of a canvas area on which the user can click to place a number. When using unconventional GUI components, however, it is likely that the replay process will get stuck.

The MoveIt-M application (Bescherer et al., 2011) was originally incompatible with Jacareto because applets were embedded in its GUI, a rare combination that triggered a previously unknown Jacareto bug. Even though that bug has been fixed, there are still problems because Jacareto cannot handle animations that are part of MoveIt-M. In other cases, replay quirks were caused by bugs in the Java platform itself. SetSails! playback suffered from a Java bug concerning popup menus. As this bug only affected a very small number of Java projects, the bug report was set to low priority and later closed without being resolved[2]. After tracking the problem down, a work-around had to be implemented in the Jacareto source code. Similar work had to be done for the ColProof-M application (ibid.), which was not replayable because of a bug in the drag and drop engine of Java Swing. Because of the variety of possible Java GUIs, one cannot expect a new application to be compatible with Jacareto right away without fixing or working around bugs.

Conclusion and future work

Extensions to the SetSails! learning application and to the Jacareto capture & replay tool made it possible to implement a learning scenario with semi-automatic assessment. This enables students to already get support during their solution processes. Many of the enhancements described in this contribution are also beneficial to using Jacareto with any other Java-based learning application. However, experiences have shown that it is too early to recommend Jacareto as a general-purpose assessment framework.

SetSails! and Jacareto will be evaluated during the summer term of 2011. These studies will show in which way students make use of the semi-automatic feedback offerings, whether the Jacareto recordings enable tutors to get an insight of the students' conceptual understanding, and whether this helps them to deliver elaborate feedback.

References

Abdel Nabi, D. & Rogers, P. (2009). The use of screen recorders for assessment and learning of data analysis. *Psychology, Learning and Teaching*, 8(1), 21-28.

Bescherer, C., Herding, D., Kortenkamp, U., Müller, W. & Zimmermann, M. (2011, in press). E-Learning Tools with Intelligent Assessment and Feedback. In S. Graf et al. (eds.): *Adaptivity and Intelligent Support in Learning Environments.* IGI Global.

[2] Java bug report: http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4394642 (accessed 29 April 2011)

Chandler, P. & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition and Instruction*, 8(4), 293–332.

Conole, G. & Warburton, B. (2005). A review of computer-assisted assessment. In *Research in Learning Technology*, 13: 1, 17-31. Routledge.

Guéraud, V. , Adam, J.-M., Lejeune, A., Dubois, M. & Mandran, N. (2009). Teachers need support too: FORMID-Observer, a Flexible Environment for Supervising Simulation-Based Learning Situations. In *Proceedings of the Intelligent Support for Exploratory Environments Workshop*.

Herding, D., Zimmermann, M., Bescherer, C. & Schroeder, U. (2010). Entwicklung eines Frameworks für semi-automatisches Feedback zur Unterstützung bei Lernprozessen. In *DELFI 2010: Tagungsband der 8. e-Learning Fachtagung Informatik,* 145-156.

Mason, R. & Bacsich, P. (1998). Embedding Computer Conferencing into University Teaching. In *Computers & Education*, 30(3-4), 249-258. Elsevier.

Schroeder, U. & Spannagel, C. (2006). Supporting the Active Learning Process. *International Journal on E-Learning, 5*(2), 245-264.

Spannagel, C. & Kortenkamp, U. (2009). Demonstrating, Guiding, and Analyzing Processes in Dynamic Geometry Systems. In *Proceedings of the 9th International Conference on Technology in Mathematics Teaching*.

Spannagel, C. (2007). *Benutzungsprozesse beim Lernen und Lehren mit Computern*. Hildesheim, Berlin: Franzbecker.