**E-learning Assessment Tool**

Radu Adrian Cioră¹, Carmen Simion² and Claudiu Voinia³

Lucian Blaga University of Sibiu, radu.ciora@ulbsibiu.ro
Lucian Blaga University of Sibiu, carmen.simion@ulbsibiu.ro
Lucian Blaga University of Sibiu, claudiu.voinia@ulbsibiu.ro

**ABSTRACT:** Learner to computer assessment is an important part of the educational process. It is the final step of the education process. We present a novel approach to computer based knowledge assessment. Our system is incorporated into Windows and monitors the learner’s actions. The user is given a specific task to perform and the computer analyzes the actions that he performs and compares them against an action list that needs to be fulfilled for that particular task to be completed. If all the actions are executed correctly and in the necessary order, then we say that the student has successfully completed the task. If the learner fails to complete the task he will be provided with feedback, specifying the steps where he has failed.

**KEY WORDS:** e-learning, assessment, ontology.

1. **INTRODUCTION**

We propose a computer driven e-learning assessment tool. Our system is able to assess a student without any third party interference. Until now, learner assessment was question-answering based or by using multiple choice questionnaires. Our system is able to assess a student while he performs a specific task that he is given. Based on the actions the learner performs, the system decides if he successfully completed the task or not. Therefore the burden of a supervisor assessing a learner is reduced with the aid of our tool. Our system monitors the student while he performs a given task. It records his actions and compares them with a list of predefined actions that has to be fulfilled in order to complete a certain task that he is given. If the two actions lists match, then we say that the task was completed successfully, otherwise we say that the task was not successfully completed and the system show the learner the actions where he failed and a way how he could have completed the task successfully.

2. **BACKGROUND AND CONTEXT**

The existing work in JELFAD project tries to demonstrate the operation of their e-Learning Framework tools and content that could be evaluated in a live student assessment context. A second aim of the project consisted in evaluating the success with which outputs from sequencing or learning projects can be integrated with the outputs from assessment projects.

The ASAP project produced a set of tools for student assessment. It consisted of “Automatically Generated Questions” and the “Student Generated Response Questions”. The latter are evaluated using an automated marking system currently using a simple XML response protocol.

The TreCX project is an open source toolkit that is able to perform tracking of learners across distributed e-learning systems. It allows tracking data to be collected from a variety of e-learning applications and then presented back as a coherent whole so that the actions of a learner can be followed across application boundaries. TreCX contains a standalone web-based tracking event store that can be used by e-learning applications to store the actions of learners in an XML format; a publishing library that enables an existing application to push events to the tracking store using a REST interface. A reporting module with a REST interface exists for facilitating the creation of reporting applications that can interrogate one or more tracking store instances.

3. **PROBLEM**

Until now, assessment tools did not provide a mechanism for hands-on, “live” knowledge testing. There are no automated assessment tools able to assess a learner while he performs a certain task on a computer. The only automated assessment solutions available out on the market are multiple choice questionnaires, which may contain some degree of controlled randomness in the generated questions. For example, let’s take the case of a student who is requested to send an e-mail to ‘Foo.Bar@ulbsibiu.ro’, using Microsoft Outlook, with the subject: ‘my test e-mail’ and an empty body. With a questionnaire you would have a question like the following:
Which of the following actions is a user required to do in order to send an e-mail:

a) Start Microsoft Outlook
b) Click the new e-mail button
c) Fill the To field
d) Fill the Cc field
e) Fill the Subject field
f) Write the e-mail text
g) Click the send button

The correct answers would be a, b, c and g. But what if you want to test whether the learner “really” knows how to send an e-mail or not. Ordinarily for this use-case you would need to give the student a computer and have a teacher monitor him while he sends the e-mail. This is a waste of time for the teacher, taking into account the fact that he has to pass by every student’s computer and observe the same procedure over and over again. This is actually the role of computers in today’s society: to perform such repetitive tasks. Our approach successfully makes the teacher’s task of assessing the learner easier, by providing him/her with a live assessment tool, which we detail in the next section.

4. PROTOTYPE IMPLEMENTATION AND USE CASE TESTING

The initial phase was the design of a tracking tool able to realize the actions of a computer user and be at the same time context-aware [8]. We developed an ontology-driven tracking tool able to infer the actual actions that a user was doing on a computer based on knowledge the currently focused application and the input that he fed into the application – mouse clicks and scrolls or keyboard’s key presses. The ontology called “context” models the computer-human interaction. The ontology has three main branches:

- The interface branch – which maps the input events raised by the input devices (mouse & keyboard); for example we map the 522 and 256 interrupts which correspond to mouse scroll and key down events in Windows.
- The application branch – which creates a hierarchy of applications based on the application type (ex. Document editor – Microsoft Word, e-mail client – Microsoft Outlook).
- The activity branch – is the main branch in the ontology, it maps the activities of a user based on both the active application and interface currently being used for a certain task. For instance, if the Microsoft Outlook application is active and the 522 mouse scroll event occurs then we can create associated triples to add to our graph to log the application - Microsoft Outlook and the event - Mouse scroll. The resulting graph provides a picture of the user’s activities over time and allows interesting queries to be performed on it using SPARQL [11] to provide views from numerous angles. For our special case the resulting graph contains just one node with the value: “User reads an email in Microsoft Outlook”.

We designed the ontology in Protégé and exported it in RDF/RDFS format, which we considered to be the most appropriate for our needs. As the ontology is not able to map every aspect of the user’s actions, we used three types of interfaces in order to be able to get the actions that a user performs on a PC.

The first interface we used was the OLE automation interface. This interface is based on the Component Object Model (COM). COM is a standard software architecture based on interfaces that is designed to separate code into self-contained objects, or components. Each component exposes a set of interfaces through which all communication to the component is handled. The OLE automation interface’s primary goal is for automating applications. We are not interested in this part of the interface, but rather on the information that it can provide about the applications it connects us to. This information goes from the Word document’s author to the subject or the content of an e-mail.

A second interface we used was the hooking interface. A hook is a point in the system message handling mechanism where an application can install a subroutine to monitor message traffic in the system and process certain types of messages. The system supports many types of hooks, but our interest was limited in hooking the mouse and the keyboard. This interface gave us access to the raw input of the input devices; this was useful at some extent - meaning that we could get what, where and when the user clicked something, but it would not give us the information which UI element was activated.

For being able to tell which UI element was activated we need the aid of a third API called Active Accessibility. The Active Accessibility is meant to help accessibility aids, here our application, interact with UI elements of other applications and the operating system. So this interface allows us to tell which UI element a user activated at a given time.
So with the aid of these three interfaces we are able to capture all the actions a learner does on a computer. We extended the tool with a logging functionality, so we could keep a history of the student’s actions. On top of this low-level tracking tool we started building an assessment tool, which has as input an XML document. The XML document contains the description of a task that a student has to execute. It can be easily created using a graphical tool that we implemented. A sample of such an XML document can be seen in Figure 1. Our program sits in background and monitors the learner’s actions. When the student closes the application we present him with the outcome. If the he executes the tasks correctly – meaning that he completed all the necessary steps in the required order, the system tells the learner that he successfully completed the test. If he failed to do so, the system provides him with feedback on the errors he/she committed. It shows him the steps that he failed and provides an explanation why he failed.

While development is still ongoing, initial experiments in assessing the learner have shown some positive results which make this tool widely applicable. We created a set of use cases based on real world scenarios that is able to test our prototype. If we take the use case of sending an e-mail, it exposes the overall workflow of the automated assessment in our system. As our approach is context aware the monitoring system is smart enough not to overload the system with unnecessary processing, by filtering the irrelevant events. In this way the tracking ignores the events that are not relevant to the current assessed task, for example if a user switched context (switched to a different application), while he was in the middle of a task and started doing something else, the system no longer processes his/her actions, but will resume monitoring as soon as he switches back to the task given.

Figure 1. Live Assessment with Learning Help and Simulation.
5. CONCLUSIONS AND INTENXTIONS

We have described a system that automatically assesses a user and provides him with feedback, without any human interaction. Previous systems treat the problem of assessment using multiple choice questionnaires or were question-answering based.

We believe that a “hands-on” assessment tool is much more engaging and more educative than the conventional approaches. The easiness of extending the system either with new tests(tasks) – based on XML input or to cover new applications – based on the ontology, makes it very scalable.

In future we want to extend the system to be able to assess multiple tasks at the same time and even cross/multi-application tasks. Additionally, we intend to implement a simulation module which would show a student the necessary steps that need to be performed in order to fulfill a task.

6. ACKNOWLEDGEMENTS

We gratefully acknowledge S.P. Williams for his valuable feedback. This work was supported by Sectoral Operational Programme Human Resources Development 2007-2013 Priority Axis 1: “Education and training in support for growth and development of a knowledge based society”, key area of intervention 1.5: “Doctoral and post-doctoral studies programmes for research support”. Project title: “Harmonization of the Romanian academic valences of the European Community”. Contract code: POSDRU/CPP107/DMI1.5/S/76851. Beneficiary: Lucian Blaga University of Sibiu.

7. REFERENCES

► DTI, (2006), Beyond eLearning: practical insights from the USA.
► Moran, T.P., (1994), Context in Design. Human-Computer Interaction, (Special Issue on context in design.).