Logging and Visualization of Learner Behaviour in Web-Based E-Testing

Gennaro Costagliola, Vittorio Fuccella, Massimiliano Giordano, Giuseppe Polese
Dipartimento di Matematica e Informatica – Università degli Studi di Salerno, Fisciano(SA),Italy
{gcostagliola, vfuccella, mgiordano, gpolese}@unisa.it

ABSTRACT
In this paper we present a system for the logging and the visualization of the learners' behaviour during the execution of structured tests based on Multiple Choice question type. The system is composed of two main components: a logging framework which, instantiated in e-testing systems, produces an XML formatted log of learners' interactions with the system during tests and a stand-alone application which visualizes charts containing a chronological review of the tests. By analyzing the charts obtained through an experiment led in our department, we have defined several typical strategies used by the learners to execute tests. The effectiveness of these strategies has been inferred by correlating the strategies with the obtained scores. Further useful applications of our system allow us to detect correlations among questions and cheating attempts by the learners.

Keywords
e-testing, computer assisted assessment, CAA, on-line testing, test, learner interactions, multiple choice, learner behaviour, information visualization.

1. INTRODUCTION
E-testing systems are more and more widely adopted in academic environments combined with other assessment means. Through these systems, tests composed of several question types can be presented to the learners in order to assess their knowledge. *Multiple Choice* question type is extremely popular, since, among other advantages, a large number of its outcomes can be easily corrected automatically.

Among the disadvantages of structured tests, a low acceptance of the exam type by the learners is rather frequently noticed: many learners are afraid of not being able to best express their capacity, due to the characteristic of *multiple choice* questions of being *closed*. Even many examiners wonder if structured tests are effective in assessing the learners' knowledge and if some learners are conditioned more by the question type than by the actual question difficulty.

In order to teach to the learners how to better perform on structured tests, besides giving the usual advice of enhancing their knowledge by studying harder, and of devoting some time on practising with structured tests, several experiments aimed at tracking learners' behaviour during structured exam tests based on multiple choice question items have been carried out in the past.

Some of them [2, 3] have regarded the learner's habit to verify the given answers and to change them with other, more plausible, options. In the experiments, *right-to-wrong* and *wrong-to-right* changes have been recorded and their number has been correlated to learners' final mark on the test. Similar experiments have been carried out by correlating the time to complete the test with the final mark. [10, 15].

An interesting experiment [13] focuses on the strategies used by several learners with different abilities ("A", "C" and "F" students) to complete the tests. In particular, the frequency of several habits has been recorded and then correlated to learners' final mark on the test, such as: the habit of giving a rash answer to the question as soon as a plausible option is detected, often neglecting to consider further options; the habit of initially skipping the questions whose answer is more uncertain, in order to evaluate them subsequently; etc.

The above experiments only consider some aspects of the execution of a test. We have not found in literature any study which considers the learner behaviour as a whole. In this paper we use information visualization in order to define several typical strategies used by the learners to execute tests. When exploring data, humans look for structures, patterns and relationships between data elements. Such analysis is easier if the data are presented in a graphical form in a visualization. Information visualization is defined as the use of interactive visual representation of abstract data to amplify cognition [17]. In the past, information visualization has successfully used in an e-learning application to measure the participation of the learners to on-line activities [11].

Our technique consists of logging all the interactions of the learners with the e-testing system interface. To elaborate, we capture the occurrence of question browsing and answering events by the learners. These data are used to visualize charts containing a chronological review of the tests. Besides identifying the most employed strategies, we try to determine their effectiveness by correlating their use with the scores obtained on the tests. Another useful application of our system allows us to

detect correlations among questions: if a question contains a suggestion to solve other questions, this is easily visible in the charts. Lastly, unethical behaviours from the learners, such as cheating by watching on others' screens and *gaming the system* [1] attempts can be detected.

The data acquired using our system can be regarded as reliable as possible, since it allows us to record information about learners' habits during on-line tests without informing them of the experiment and, consequently, without asking them to modify their behaviour, thus obtaining experiments more realistic than those performed by exploiting the "think out loud" method on *papery* tests. Our system uses the AJAX [14] technology, acronym of *Asynchronous JavaScript and XML*, which allows Web-based applications to gain more interactivity. It is the use of AJAX technology which allows us to reach our aim of capturing all of the learner's interactions with the e-testing system interface (running in the Web browser) during the test.

A fundamental component of our system is a logging framework which, once captured the client-side interactions, sends them to a server-side module. The latter records them in a suitable XML formatted log. Our framework can be instantiated on any e-testing system developed with Java technology. Another fundamental component of our system is the stand-alone application which allows us to extract the data from the log and to visualize them.

In order to demonstrate the effectiveness of our system, it has been used in the ambit of a university course at our department: the framework has been instantiated in an existing e-testing system, *eWorkbook* [5], which has been used to administer on-line tests to learners. The grade obtained on the tests has concurred to determine the final grade of the course exam.

The rest of the paper is organized as follows: section 2 gives the knowledge background necessary to understand some concepts on which the system is based. The logging framework and its integration in *eWorkbook* is presented in section 3. Lastly, in section 4, we discuss the techniques employed and the results obtained through the experiments. Several final remarks and a brief discussion on future work conclude the paper.

2. BACKROUND: THE AJAX TECHNOLOGIES

AJAX is a style of web application development that uses a mix of modern web technologies to provide a more interactive user experience. *AJAX* is not a technology. It is an approach to web applications that includes a couple of technologies. These are *JavaScript*, *HTML*, *Cascading Style Sheets (CSS)*, *Document Object Model* (*DOM*), *XML* and *XSLT*, and *XMLHttpRequest* as a messaging protocol.

These core technologies are mature, well-known and widely used in web applications. *AJAX* became so popular because it has a couple of advantages for the browser based web applications developers. It eliminates the stop-start nature of interactions, user interactions with the server happen asynchronously, data can be manipulated without having to render the entire page again and again in the web browser, and requests and responses over the *XMLHttpRequest* protocol are structured *XML* documents. This enables developers easily integrate *AJAX* applications into Web Services.

AJAX drew some attention in the public after *Google* started developing some new interesting applications. Some of the major products Google has introduced over the last year by using the *AJAX* model are *Google Groups*, *Google Suggests*, *GMail*, and *Google Maps*. Besides the *Google* products *Amazon* also has used the *AJAX* approach in its search engine application.

A developer can use *AJAX* in his/her web applications by just writing his/her own custom JavaScript code that directly uses the *XMLHttpRequest* protocol's *API*. However, the developer must take into consideration the implementation differences among the web browsers. Instead of using pure *AJAX* and dealing with the browser differences, the developers can use some newly developed libraries which provide higher level *AJAX* services and hide the differences between browsers. Among these are *DWR*, *Prototype*, *Sajax*, and *AJAX.NET*.

3. THE LOGGING FRAMEWORK

The purpose of the Logging Framework is to gather all of the learner actions during the browsing of the Web pages of the test and to store raw information in a set of log files in XML format.

The framework is composed of a server-side and a client-side module. The client-side module is responsible for "being aware" of the behaviour of the learner while he/she is browsing the test pages. The server-side module receives the data from the client and creates and stores log files on the disk.

Despite the required interactivity level, due to the availability of AJAX, it has been possible to implement the client-side module of our framework without developing plug-in or external modules for Web browsers. Javascript has been used on the client to capture learner interactions and the text-based communication between the client and the server has been implemented through AJAX method calls. The client-side scripts are added to the e-testing system pages with a light effort by the programmer.

The events captured by the framework are the following:

- Actions undertaken on the browser window (open, close, resize, load, unload)
- Actions undertaken in the browser client area (key pressing, scrolling, mouse movements and clicks)

The event data is gathered on the browser and sent to the server at regular intervals. It is worth noting that the event capture does not prevent other scripts present in the page to run properly.

The server-side module has been implemented as a Java servlet which receives the data from the client and prepares an XML document in memory. At the end of the test session the XML document is written to the disk. The logger can be instantiated and then enabled through the configuration.

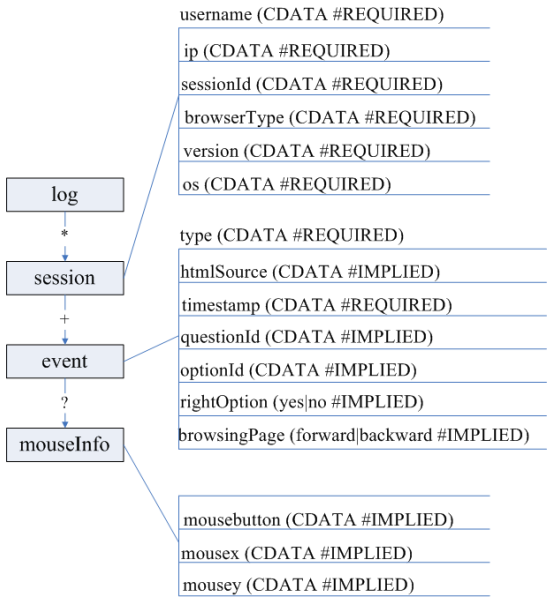

Figure 1. The information model for log data

The information model used for the log data is quite simple and is shown in figure 1. The information is organized per learner test session. At this level, the username (if available), the IP of the learner and session identifier are logged as well as agent information (browser type, version and operating system). Inside a session, a list of event elements is present. The data about the user interactions are the following:

- Event type
- HTML source object involved in the event (if present)
- Mouse information (pressed button, coordinates)
- Timing information (timestamp of the event)
- More information specific of the event. I.e. for a response type event (a response given to a question), the question and option identifiers and the indication whether the response was right or wrong are recorded.

An important concern in logging is log size. If an experiment is done involving a large set of learners and the test is composed of many questions, log files can reach big sizes. A configuration system, including the following configuration settings, has been conceived in order to reduce log sizes:

- List of events to capture
- Sub-set of attributes to store in the log for each event
- Sections of the Web pages (*divs* or table cells) to monitor as sources of the events
- Time interval between two data transmissions from the client to the server
- Sensitivity for mouse movements (short movements are not captured)

The configuration is read by the server-side module but affects the generation of the javascript modules running on the client-side. The architecture of the framework is graphically represented in figure 2.

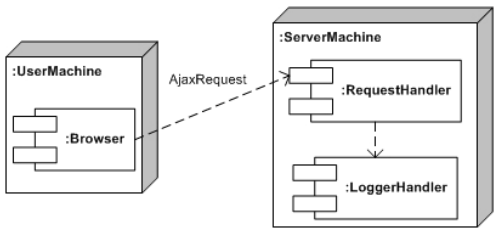

Figure 2. The Logging Framework Architecture

On the client machine, everything can be done in the web Browser. The Javascript modules for event capturing, dynamically generated on the server according to the configuration settings, are downloaded and run in the browser interpreter. Data is sent to the server through an *AJAX* request. On the server-side, a module called *RequestHandler* receives the data and sends it to a module called *LoggerHandler*, which organizes the XML document in memory and flushes it to the disk every time a learner test session finishes.

3.1 The eWorkbook System and Instantiation of the Framework

The above described framework has been instantiated in an existing Web-based e-testing system, named *eWorkbook* [5], which can be used for evaluating learner's knowledge by creating (the tutor) and taking (the learner) on-line tests based on multiple choice question types.

The questions are kept in a *hierarchical* repository, that is, the repository is tree-structured, in the same way as the file system of an operating system. In this structure, the files can be thought of as questions, while the directories can be thought of as *macroareas*, which are containers of questions usually dealing with the same subject.

The tests are composed of one or more *sections*. This structure facilitates the selection of the questions from the database, but it is still useful for the assessment, where it can be important to establish if one section is more important than another to determine the final grade for the test. There are two kinds of sections: *static* and *dynamic*. The difference between them is in the way they allow question selection: for a *static* section, the questions are chosen by the tutor. For a *dynamic* section, some selection parameters must be specified, such as the difficulty, leaving the system to choose the questions randomly whenever a learner takes a test. In this way, it is possible with *eWorkbook* to make a test with banks of items of different difficulties, thus balancing test difficulty, in order to better assess a heterogeneous set of learners.

eWorkbook adopts the classical *three-tier* architecture of the most common *J2EE* Web-applications. The *Jakarta Struts* framework has been used to support the *Model 2* design paradigm, a variation of the classic *Model View Controller* (*MVC*) approach. Struts provides its own *Controller* component and integrates with other technologies to provide the *Model* and the *View*. In our design choice, *Struts* works with *JSP*, for the *View*, while it interacts with *Hibernate* [9], a powerful framework for *object/relational persistence* and query service for Java, for the *Model*.

The application is fully accessible with a Web Browser. Navigation is facilitated across the simple interfaces based on menus and navigation bars. User data inserting is done through HTML forms and some form data integrity checks are performed using Javascript code, to alleviate the server side processes. A big effort was made to limit the use of client-side scripts to the standard *ECMAScript* language [7]. No browser plug-in installations are needed. It is worth noting that the system has been tested on recent versions of the most common browsers (i.e., *Internet Explorer*, *Netscape Navigator*, *Firefox* and *Opera*).

The integration of the server-side component in the *eWorkbook* system has been rather simple: the *JAR* (*Java ARchive*) file containing the framework classes has been imported as a library in the system. A modification to the system's deployment descriptor has been necessary in order to deploy the server-side module (*servlet*) which receives the events from the client.

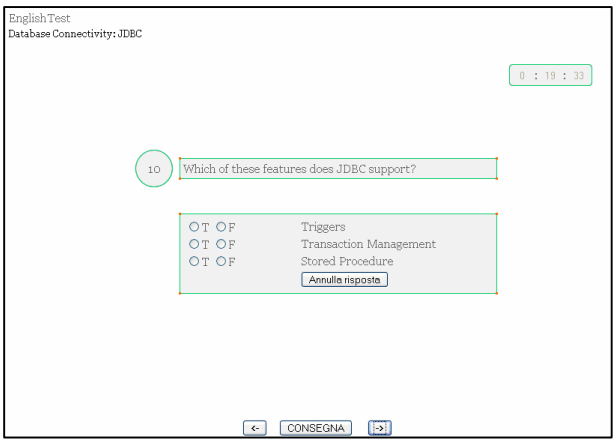

Figure 3. A Screenshot of the Test Execution

The integration of the client-side component of the framework, composed of several *Javascript* files, in the system has been slightly more complicated, due to the structure of *eWorkbook*'s interface: the test is launched in a child browser window of the main system Web page. A screenshot of the full-screen window containing the test is shown in figure 3. This window displays a timer to inform the learner of the remaining time to complete the test and contains the controls to flow among the questions (forward and backward buttons) and the button to submit the test. The stem and the form containing the options are loaded in an iframe window present in the centre of the page.

The javascript modules for capturing the events have been included in both the external window and the internal frame, while the modules for sending the events to the server have only been included in the page loaded in the external window. HTML element identifiers have been inserted in the table cells which format the form elements containing the options and the respective *radiobuttons*. This task has been done in order to identify the source element of the events.

4. TEST VISUALIZATION

The system has been experimented by using it across a test session in a university course: *eWorkbook* has been used to administer on-line tests to learners. The learners were not informed of the experiment; they just knew that the grade obtained on the tests concurred to determine the final grade of the course exam.

The test, containing a set of 25 items to complete in a maximum time of 20 minutes, was administered to 71 learners, who took the test concurrently in the same lab. The assessment strategy did not foresee any penalty for incorrect responses and the learner were aware of that.

The logger was enabled and an approximately 4Mb sized *XML* log file has been obtained. The logging activity produced no visible system performance degrading.

The next sub-section shows the chart production phase, while the subsequent sub-sections describe the experiments performed using log data.

4.1 Chart Production

A chronological review of the test has been made available through a chart, obtained by showing the salient points of a test execution, synthesized in the interactions recorded in the log file. This chart shows, at any time, the item browsed by the learner, the mouse position (intended as the presence of the mouse pointer on the stem or on one of the options) and the presence of response type interactions, correct or incorrect.

The chart is bydimensional. The horizontal axis reports a continuous measure, the time, while the vertical axis displays categories, the progressive number of the item currently viewed by the learner.

The test execution is represented through a broken line. The view of an item for a determined duration, is shown through a segment drawn from the point corresponding to the start time of the view to the one corresponding to its end. Consequently, the length of the segment is proportional to the duration of the visualization of the corresponding item. A vertical segment represents a browsing event. A segment oriented towards the bottom of the chart represents a backward event, that is, the learner has pressed the button to view the previous item. A segment oriented towards the top is a forward event. Using the logger with eWorkbook, we will only see one unit long segments (except for the transition from the last to the first question), since *eWorkbook* only allows to browse items sequentially.

The responses given by a learner on an item are represented through circles. The progressive number of the chosen option is printed inside the circle. The indication of correct/incorrect response is given by the filling colour of the circle: a blue circle represents a correct response, while an incorrect response is represented through a red circle.

The colour is used also for representing the position of the mouse pointer during the item view. The position of the mouse pointer can be a significant indicator of the part of the item analyzed by the learner, since it has been demonstrated to be correlated to the position of the gaze of the user [4]. The presence of the mouse pointer in the stem area is represented through a black colour for the line. As for the options areas, the red, yellow, green, blue and purple colours have been used, respectively, for 1 to 5 numbered options. More options are not supported. At last, grey is used to report the presence of the mouse pointer in a neutral zone.

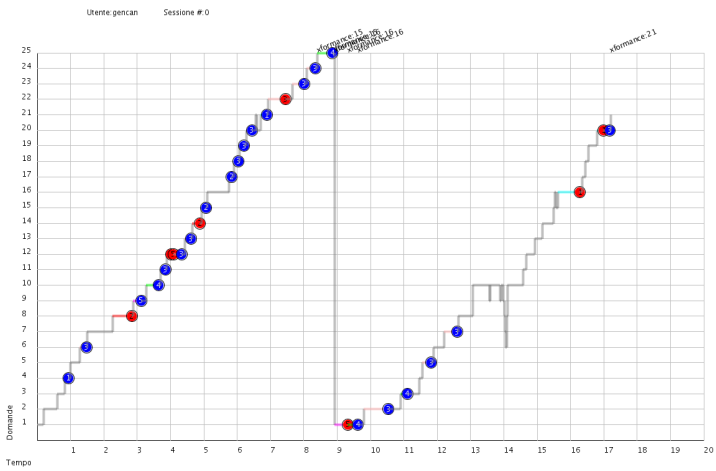

Figure 4. Graphical Chronological Review of a Sample Test

The graphical chronological review of a sample test is shown in figure 4. The test is composed of 25 items and the maximum duration foreseen is 20 minutes, even if the learner has completed and submitted the test in 17 minutes approximately. The whole view of the chart in the figure

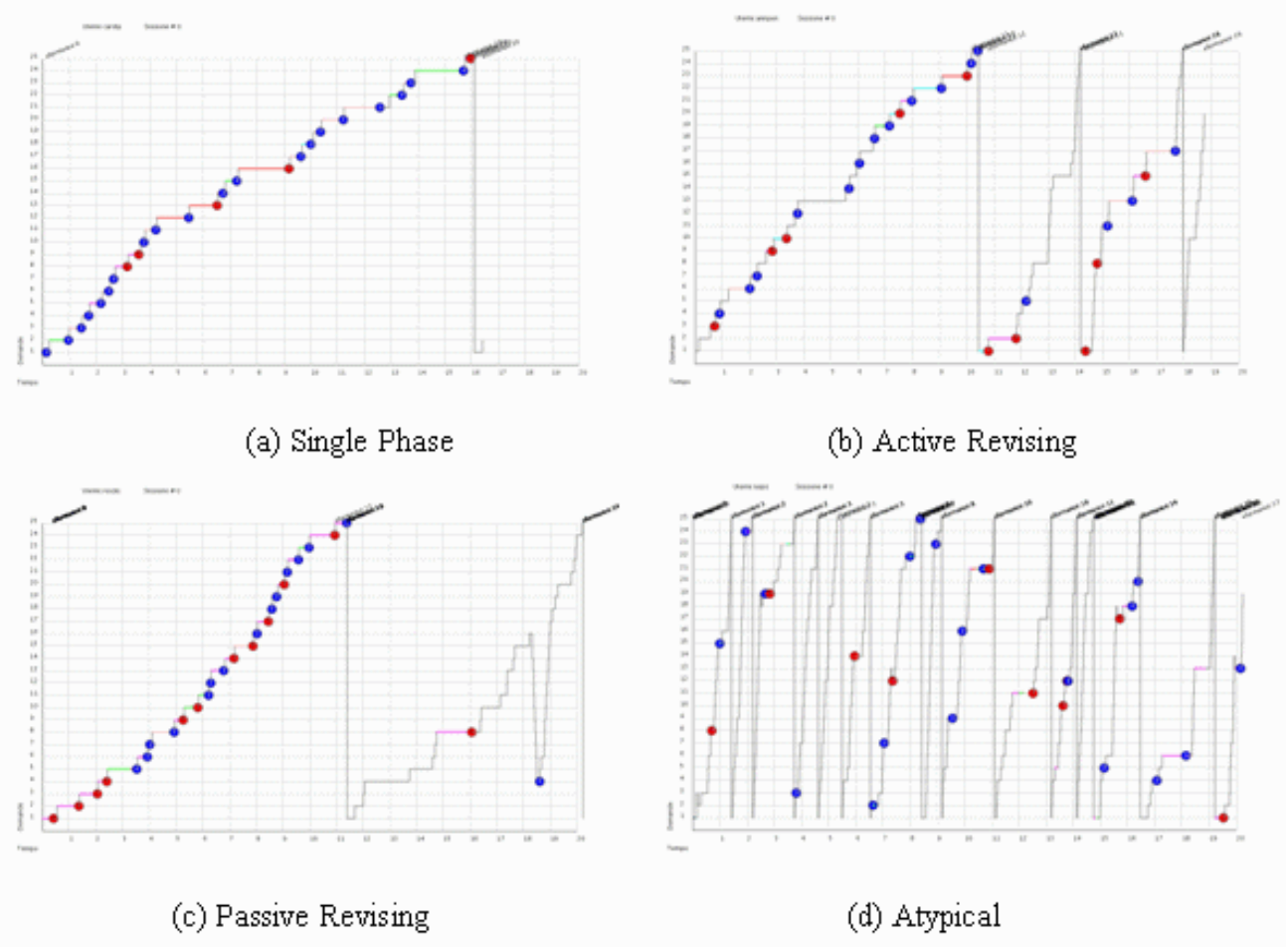

Figure 5. Samples of Test Execution Strategies

shows the strategy adopted by the learner in the test: the test execution is visibly composed of two successive *phases*. In the first one (nearly 9 minutes), the learner has completed the view of all the items from 1 to 25. Responses to 19 questions have been given in this phase. Several items present more than one response, maybe due to learner's reflection, while a few items have been skipped. The mouse position for each item view is visible through a zoom, performed by using a suitable zoom tool of the analyzer.

4.2 Strategies for Executing Tests

From the analysis of the charts obtained through our experiment the following strategies, with a few exceptions, have been adopted by the students to execute the tests:

- **Single Phase**. This strategy is composed of just one *phase*. The time available to complete the test is organized by the learner in order to browse all the questions just once. The learner tries to reason upon a question for an adequate time and then gives a response in almost all cases, since he/she knows that there will not be a revision for the questions. Eventual *phases* subsequent to the first one have a negligible duration and no responses. A sample of this strategy is shown in figure 5a.
- **Active Revising**. This strategy is composed of two or more *phases*. The learner intentionally browses all the questions in a shorter time than the time available, in order to leave some time for revising *phases*. The questions whose answer is uncertain are skipped and the response is left to subsequent *phases*. As a general rule, the first *phase* lasts a longer time and the subsequent *phases* have decreasing durations. A sample of this strategy is shown in figure 5b.
- **Passive Revising**. This strategy is composed of two or more *phases*. The learner browses and

answers all the questions as fast as possible. The remaining time is used for one or more revising *phases*. As a general rule, the first *phase* lasts a longer time and the subsequent *phases* have decreasing durations. A sample of this strategy is shown in figure 5c.

For both the definition of the strategies and the classification of test instances, the charts have been visually analyzed by a human operator. The above tasks are rather difficult to perform automatically, while a trained human operator can establish the strategy used by the learner from a visual inspection of the charts of the test instances and giving advice to the learners on how to perform better next time.

According to the data of our experiment, the most frequently adopted strategy is the *Active Revising*, used by 40 learners over 71 (56,5 %), followed by the *Passive Revising* strategy (20 learners over 71, 28,2%) and by the *Single Phase* one, used only in 9 cases over 71 (12,7%). Only two learners have adopted an atypical strategy (see figure 5d) which cannot be classified in any of the previously described patterns.

The best results have been obtained by the learners who adopted the *Passive Revising* strategy, with an average score of 17.6 exact responses on the 25 questions test. With the *Active Revising*, instead, an average score of 16.4 has been obtained. Lastly, the *Single Phase* strategy reveals itself as the worse one, showing an average score of 15.1.

Then, a winning strategy foresees more than one *phase* and this is confirmed by the lightly positive linear correlation (0.14) observed between the number of *phases* and the score obtained on the test. For both the strategies that foresee more than one phase, the score is often improved through the execution of a new *phase*. The improvement is evident in the early *phases* and tends to be negligible on the growing of the phase number: starting from a value of 14.3 obtained after the first *phase*, the average score increases to 16.2 after the second *phase*. The presence of further *phases* brings the final score to an average value of 16.5. The average duration of the first *phase* of the *Passive* strategy (14'50") is longer than the one registered for the *Active* strategy (11'51"). This result was predictable, since, by definition, the *Active* strategy foresees the skipping (= less reasoning) of the questions whose answer is more uncertain for the learners. Another predictable result, due to the above arguments, is that the *Passive* strategy has less *phases* than the *Active* one, on average (2.55 and 3.2, respectively).

4.3 Detection of Cheating

As proven by several studies in the education field, many learners cheat at exams, when they can [6, 8]. Cheating detection in assessment tests is not an easy task: most of the techniques employed so far have been based on the comparison of the results obtained in the tests [12]. These techniques cannot give the certainty of the guilt, since a high similarity of two tests can be due to coincidence. Furthermore, as in all fraud detection systems, the task is complicated by several technological and methodological problems [16]. It could be useful to gain information on the learners' behaviour during the test. Analysis on these data can be integrated to results comparison in order to have a more comprehensive data set as input for a data mining technique to detect cheating. For example, let's consider the following situation: during the test, *learner A* answers *true* to a question and *learner B*, who is seated behind the former, answers the same few instants later. The tracking of this information, available through the charts of our system, could be useful to prove that the learner has cheated, looking on the screen of his classmate.

Copying from the colleague set in front of someone is not the only one frequently encountered cheating exploit: in some cases several attempts of *gaming the system* [1] have been reported. This exploit consists of executing a large number of tests with the only scope of having from the e-testing feedback system as much correct responses as possible. When some suspect cases are detected from the frequency of self-assessment test access and the *strange* scores obtained, our system can confirm the cheating attempts, by revealing atypical patterns of test execution.

4.4 Detection of Correlation Among Questions

By visually inspecting learner's behaviour we can also be assisted in the detection of correlated questions. In some cases, a visual pattern similar to a *stalactite* occurs in the chart, as shown in figure 6.

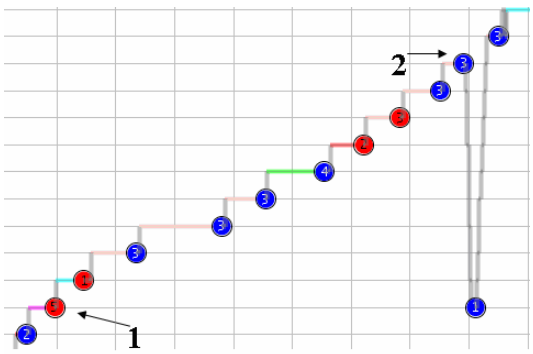

Figure 6. A Stalactite in the Chart Shows the Correlation Between Two Questions

The occurrence of such a pattern tells that, while the learner was browsing the current question, he/she could deduce the right answer to a previous question. In the example shown in the figure, the learner who was browsing the question "2", understood that he/she had given a wrong response to the question "1", and came back to change the response, from option 5 to option 1.

5. CONCLUSION

In this paper we have presented a system for capturing and visualizing the behaviour of the learners during e-testing sessions based on structured tests. The system is composed of a logging framework which can be instantiated in e-testing systems and of a stand-alone application which produces the charts. A chart produced with our system shows the chronological review of a test executed by a learner.

The use of information visualization in this context has been proven to be useful for various applications, such as analysis of the strategies used by the learners during the execution of a structured test; cheating detection and detection of correlation among questions. We are confident that more interesting applications can be discovered. Furthermore, our system allows experimenters to perform realistic experiments, since learners are not informed of the experiment and are not forced to modify their behaviour.

The system has been used for an experiment under determinate circumstances (established number of options per item, time to complete the test adequate to the number of questions, assessment strategy known by the learners and with no penalty factors). Future work is aimed at performing new experiments in more general conditions. Further studies will also be devoted to understand if valuable information can be obtained by observing the mouse pointer position during test browsing, which is at present recorded by the log and displayed with different line colours in the chart, but has not yet been linked to any concrete application.

6. REFERENCES

[1]. R. S. Baker, A. T. Corbett, K. R. Koedinger, A. Z. Wagner. Off-task behavior in the cognitive tutor classroom: when students "game the system". *Proceedings of the 2004 conference on Human factors in computing systems*. 383 - 390

[2]. J.A. Bath, "Answer-changing Behaviour on objective examinations", *The Journal of Educational Research*, 1967, 61, 105-107.

[3]. J.B. Best, "Item difficulty and answer changing", *Teaching of Psychology*, 1979, 6, 228-240.

[4]. M.C. Chen, J.R. Anderson, M.H. Sohn Moore, "What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing.", In *CHI '01 extended abstracts on Human factors in comp. syst.* 2001

[5]. G. Costagliola, F. Ferrucci, V. Fuccella, F. Gioviale, "A Web Based Tool for Assessment and Self-Assessment", *Proceedings of ITRE '04*, London, UK, 2004, pp. 131-135

[6]. M. Dick, J. Sheard, C. Bareiss, J. Carter, D. Joyce, T. Harding, C. Laxer, Addressing student cheating: definitions and solutions, *Proc. of ITiCSE 2002*, 172-184

[7]. ECMAScript, Standard ECMA-262, *ECMAScript Language Specification*, http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf, 2007

[8]. T. S. Harding, D. D. Carpenter, S. M. Montgomery, N.H. Steneck. The current state of research on academic dishonesty among engineering students, *Proc. of FIE '01*, vol. 3, F4A 13-18

[9]. Hibernate. http://www.hibernate.org, 2007

[10]. J.J. Johnston, "Exam Taking speed and grades.", *Teaching of Psychology*, 1977, 4, 148-149

[11]. R. Mazza, V. Dimitrova, Student tracking and personalization: Visualising student tracking data to support instructors in web-based distance education, *Proceedings of the 13th Int. World Wide Web Conf. on Alternate track papers & posters*, 154 - 161

[12]. S. W. Mulvenon, R. C. Turner, S. Thomas. Techniques for detection of cheating on standardized tests using SAS". *Proc. of the 26th Annual SAS Users Group Int. Conf.*, Miami, FL, 1 – 6.

[13]. L. McClain, "Behavior during examinations: A comparison of "A", "C" and "F" students.", *Teaching of Psychology*, Vol.10, No 2 April 1983

[14]. G. Murray, "*Asynchronous JavaScript Technology and XML (AJAX) With Java 2 Platform Enterprise Edition*"

[15]. C.A. Paul, J.S. Rosenkoetter, "The relationship between the time taken to complete an examination and the test score received.", *Teaching of Psychology*, 1980, 7, 108-109.

[16]. H. Shao, H. Zhao, G. R. Chang. Applying data mining to detect fraud behavior in customs declaration. *Proc. of ICMLC'02*, 1241-1244 vol.3

[17]. Shneiderman, B. and Plaisant, C.. Show Me! Guidelines for Producing Recorded Demonstrations, Proc. of 2005 IEEE Symposium on Visual Languages and Human-Centric Computing. (VL/HCC'05) 171-178