Scalable Monitoring of Student Interaction Indicators in Exploratory Learning Environments

Sergio Gutierrez-Santos
London Knowledge Lab
Birkbeck, Univ. of London
sergut@dcs.bbk.ac.uk

Stefano Capuzzi
London Knowledge Lab
Birkbeck, Univ. of London
capuzzistefano@gmail.com

Ken Kahn
London Knowledge Lab
UCL Institute of Education
toontalk@gmail.com

Sokratis Karkalas
London Knowledge Lab
Birkbeck, Univ. of London
sokratis@dcs.bbk.ac.uk

Alexandra Poulovassilis
London Knowledge Lab
Birkbeck, Univ. of London
ap@dcs.bbk.ac.uk

ABSTRACT

We present and evaluate a web-based architecture for monitoring student-system interaction indicators in Exploratory Learning Environments (ELEs), using as our case study a microworld for secondary school algebra. We discuss the challenging role of teachers in exploratory learning settings and motivate the need for visualisation and notification tools that can assist teachers in focusing their attention across the class and inform teachers' interventions. We present an architecture that can support such Teacher Assistance tools and demonstrate its scalability to allow concurrent usage by thousands of users (students and teachers).

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

exploratory learning environments, monitoring, visualisation

1. INTRODUCTION

Exploratory Learning Environments (ELEs) are a type of learning environment where the focus is on students' exploration and experimentation in a knowledge domain. ELEs range from simple games to virtual labs, complex simulators and microworlds. In order for students to benefit from interaction with such ELEs, there is recognition for the need to provide explicit pedagogical support to students [13]. This has led to research into techniques for providing personalised support to students to foster their productive interaction with ELEs [8].

The role of the teacher in an exploratory learning setting is that of a 'facilitator' or 'orchestrator' [9, 11]. However, supporting the teacher in this role poses several challenges, compounded by a learning setting in which students are each working with their own computer or handheld device. Specifically, due to the open-ended nature of the tasks that the students are working on, teachers can only be aware of what a small number of students are doing at any one time as they walk around the classroom — a typical class size in a U.K. classroom being 30 students. It is therefore hard for the teacher to know which students are progressing with the task set, who is off-task, and who is in need of additional support from the teacher beyond the feedback provided by the ELE itself. Our recognition of this difficulty has led to research into the design and deployment of *Teacher Assistance* (TA) tools to be used in conjunction with ELEs, with the aim of reducing the cognitive load on the teacher and increasing the teacher's awareness of individual students' progress and of the classroom 'state' as a whole.

Earlier work has described the design, implementation and evaluation of our TA tools [7, 12]. In contrast, the present paper addresses the question of the scalability of provision of such tools as software-as-a-service. Our case study here is the MiGen system, an intelligent ELE that aims to foster 11-14 year old students' learning of algebraic generalisation [16]. In its initial implementation, all the MiGen system components were implemented in Java and integrated into a lightweight architecture based on REST, with the aim of facilitating iterative prototyping and trialling in schools during the course of the MiGen project [18]. Our stress-testing of this implementation showed that it scales up to a few hundred users (students and teachers) working concurrently, which was sufficient for local installation and usage of the system in individual schools but not sufficient to be run nation-wide in a software-as-a-service fashion. Following the end of the MiGen project, we therefore decided to re-implement the user-facing tools as browser applications and to port the server software to Google's App Engine, in order to more fully investigate the scalability properties of the MiGen architecture.

This paper describes this implementation and the performance study conducted. The structure of the paper is as follows. Section 1 has introduced and motivated the research. Section 2 gives an overview of the context and functionalities of the MiGen system, and of related work in support for the teacher and ELEs. Section 3 describes the implementation of our architecture and Section 4 the performance study.

WWW 2016 Companion, April 11-15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2891075.

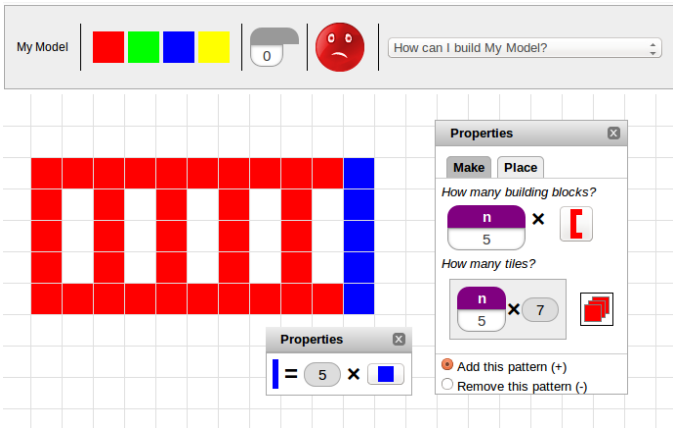

Figure 1: The eXpresser: Students create building blocks and use them to construct patterns, which they subsequently colour. They need to use variables to specify how many tiles of each colour are needed in a generalised pattern.

Section 5 gives our concluding remarks and directions of future work.

2. BACKGROUND AND RELATED WORK

2.1 The MiGen System

The MiGen project (www.migen.org) has designed and developed an intelligent, exploratory learning environment to support 11 to 14-year-old students in learning algebraic generalisation. In MiGen, students undertake tasks using a mathematical microworld called *eXpresser* — see Figure 1. These tasks ask students to create models consisting of 2-dimensional tiled, coloured patterns constructed from one or more building blocks. Firstly, specific instances of such models need to be constructed and then generalised versions in which one or more of the numbers in their construction are replaced by "unlocked" numbers, i.e. variables. In parallel, students are asked to formulate algebraic rules specifying the number of tiles of each colour that are needed to fully colour their models. The MiGen system includes an intelligent component called the *eGeneraliser* which provides both unsolicited and on-demand personalised feedback to students, based on a three-layer architecture comprising Analysis, Reasoning and Feedback Generation sub-components (see [8]).

As students work using the eXpresser on the current task they have been set, a series of *indicators* are automatically detected by the system. The indicators that are meaningful and useful for teachers in their role were identified through an iterative process undertaken collaboratively with our group of teacher collaborators on the MiGen project (see [12]). There are two categories of indicators: *task independent* (TI) and *task dependent* (TD). TI indicators refer to aspects of the student's interaction that are related to the eXpresser microworld itself and do not depend on the specific task the student is working on, e.g. 'student has placed a tile on the canvas', 'student has made a building block', 'student has unlocked a number'. In contrast, TD indicators are inferred by the eGeneraliser based on the students' actions and on knowledge specific to the current task, e.g. 'student has made a plausible building block for this task', 'student has unlocked too many numbers for this task', 'student has achieved task goal n'. We refer readers to [7] for the full list of indicator types — there are some 60 different types — and for details of how the TD indicators are inferred. All the occurrences of TI and TD indicators detected by the eXpresser or inferred by the eGeneraliser are sent to the MiGen server and are stored in the MiGen database, leading to potentially large volumes of such data.

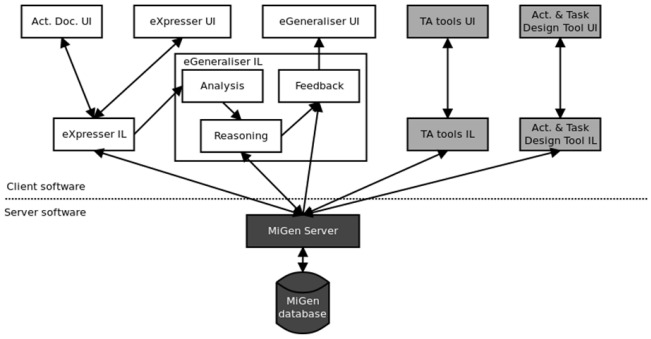

Figure 2: MiGen Logical Architecture

Figure 2 (from [16]) illustrates the logical architecture of the MiGen system. Shown in white are the components comprising the Student software running on the students' computers, in light grey the components comprising the Teacher software running on the teacher's computer, and in dark grey the Server components. Also shown is the information flow between components. Each of the user-facing tools consists of a User Interface (UI) component and an Information Layer (IL) component. Each tool's UI is responsible for interaction with the user, while its IL is responsible for managing the data structures and computation required to support the UI and for communicating with the Server software. The Server software in turn provides access to the MiGen database. 'Act.Doc.' denotes 'Activity Document', within which tasks are presented to students and they record their reflections as they undertake these tasks using eXpresser. The Act. & Task Design Tool is used to create tasks and activity documents.

In Figure 2, 'TA tools' denotes 'Teacher Assistance tools'. The TA Tools receive real-time information from the MiGen Server relating to occurrences of TI and TD indicators for each student, and each TA tool presents visually a selection of this information to the teacher. MiGen's suite of TA tools includes Student Tracking (ST), Classroom Dynamics (CD), and Goal Achievements (GA) tools (see [16, 12]). For example, Figures 3 and 4 illustrate the web browser versions of the CD and GA tools.

The CD tool gives the teacher an at-a-glance overview of which students are currently engaged with the task and who may be in difficulty and in need of help from the teacher. It represents each student by a colour-coded circle, containing the student's initials. Hovering over a circle with the cursor displays the student's full name. Clicking on a circle shows the student's current model and rule. Circles can be dragged

and moved around on the canvas, enabling the teacher to set up the display so as to match the spatial positioning of the students in the classroom. This helps the teacher to match the information being displayed in the tool with her own observations. It also helps the teacher to identify situations that may be location-dependent. For example, if several students seated near each other show as Amber this may indicate that they are distracting each other and that the teacher should intervene to refocus their attention on the task.

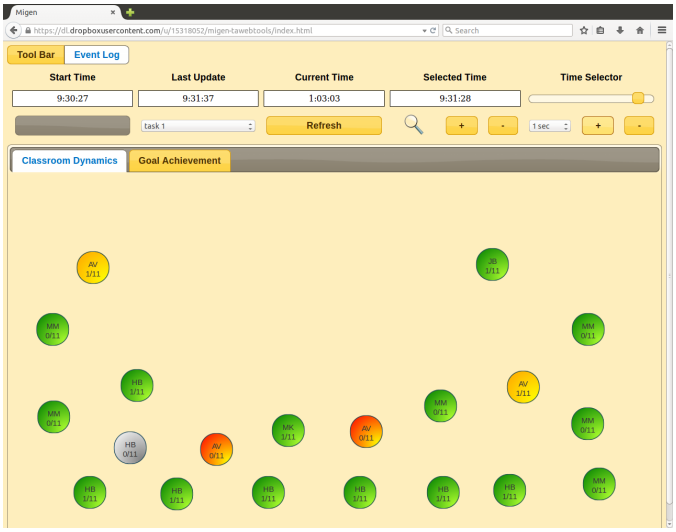

Figure 3: Class Dynamics tool. The colour of a student's circle reflects the student's current activity status, as perceived by the system. Green circles indicate students working productively on the task set. Amber circles indicate students who have not interacted with eXpresser for some time (by default, five minutes). Red circles indicate students who may benefit from immediate help. An optional feature in the CD tool shows within each student's circle the number of goals achieved so far, as a fraction of the total number of goals of the task.

The GA tool shows a tabular display of students and task goals. Each row shows the progress of one student (identified by their initials) in completing the task goals. Each column shows the completion status of one task goal across all students. Hovering over a cell with the cursor displays a full description of the goal, the name of the student, and the achievement status of that goal for that student.

2.2 Related Work

To our knowledge, MiGen's TA tools represent the first work targeted at providing teachers with information about students' progress during exploratory learning activities in the classroom (preliminary results appeared in [18, 19]).

Other early work on teacher support focused on monitoring log data generated by course management systems to help instructors' awareness of students' activities [14]; deriving statistics of students' interactions [6]; analysing system logs to help teachers understand students' behaviour in adaptive tutorials [2]; analysing CSCL synchronous interaction using rules to identify specific landmarks in the interaction [20]; and providing awareness information to teachers to support their role as moderators of multiple e-discussions [21] or class-wide collaborative activities [3].

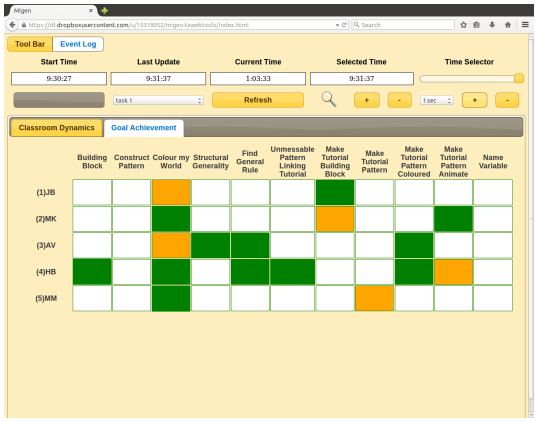

Figure 4: Goal Achievement tool. The colour each cell reflects the current achievement status of each task goal by each student, as perceived by the system. Green and white cells show whether a goal has been achieved or not. Amber shows that the goal was achieved by the student at some point during the course of the current task, but is not currently being achieved by the student's construction.

Several similar works to ours have appeared more recently, for example: Dragon *et al.* [4] build on our work in providing tools to support teachers using the Metafora system, which targets science and mathematics education; Martinez-Maldonado *et al.* [10] explore patterns of students' collaborative interactions with the aim of providing information on students' learning processes; Mercier *et al.* [15] study collaborative problem solving processes in the context of multi-touch technology. The interest in teacher support is growing also in the Learning Analytics community [5, 17, 22]. However, with the exception of a few works, e.g. [1] that visualises students' inferred plans in an ELE for chemistry, none of this work focuses on exploratory learning activities nor on the question of the scalability of provision of teacher assistance tools in such settings.

3. SYSTEM IMPLEMENTATION

The initial implementation of the MiGen system was conceived to be installed and used locally in schools. It was assumed that each school would have a copy of the MiGen server installed on their own facilities, and that clients (i.e. students' and teachers' computers) would connect to this server through the school's local area network. As mentioned earlier, the initial Java and REST-based implementation was sufficiently performant and scalable for this deployment scenario. However, to allow larger-scale dissemination and uptake of the system in schools after the end of the MiGen project, we decided to reimplement the system as a cloud-based service.

The eXpresser, eGeneraliser and server-side components of the MiGen architecture were ported onto Google's App Engine[1] (GAE) by using GWT. In particular, this trans-

[1]appengine.google.com

formed eXpresser into a web browser application. The TA tools were also transformed into a web browser application, by porting them directly onto Javascript, and connected to the server software running on Google's cloud[2].

For the new TA tools implementation, a typical 3-tier architecture was used, employing Google App Engine at the back-end as a data store. This decision was based on the premise that GAE can dynamically allocate processing power and memory space to adapt to variable workload. The amount of data that can be exchanged and the number of requests that this system can satisfy is theoretically limitless. This means that it is theoretically impossible for this part of the system to become a bottleneck in terms of scalability. For the middle tier, the cloud-based web service of Dropbox[3] was used. Although this system is not intended to be used as a web server, it proved to be very reliable, fast and convenient in terms of development, deployment, portability and administration. The fact that it does not support the deployment of server-side components did not present a problem because all the logic is implemented in the front-end tier. The client side is a relatively 'fat' browser-based application. Apart from the usual components that constitute a typical web page, we implemented a local in-memory database[4] to cache information about the tasks assigned to students, the set of students whose progress is being monitored, and the indicator occurrences generated by these students' interactions.

The data format used for data interchange in the system is XML. The indicator data sent from the MiGen client web application to the MiGen server running in the Google cloud is stored in a NoSQL datastore. A TA tools web application instance requests an update of the indicator data from the MiGen server every five seconds (by default — this is a configurable parameter). Updates are incremental: the request is parameterised according to the time stamp of the previous such request and according to the classroom ID, so that only the data required by the specific TA tools application instance is sent by the MiGen server. The data is retrieved directly from GAE in XML form. The data is then transformed into JSON through XSLT using the XML processor that is built into the browser. This data is then directly consumable by the Javascript environment without further processing. It can be directly inserted into the local in-memory database and used by the TA tools application to instantly update its visualisations.

Such updates are partial, in the sense that they do not require a full reload of the page. Ajax is used to asynchronously query GAE and incrementally update the local data structures. Apart from the obvious performance benefit, there is an additional administrative benefit that relates to the fact that there is no need to maintain the state of the TA tools user interface between the calls since there is no need to redraw the parts that do not change.

[2]The decision to use GWT for porting most of the initial implementation and pure Javascript to port the TA tools was based largely on pragmatic considerations, taking into account the development skills of the project team at the time.

[3]www.dropbox.com

[4]The JavaScript library TaffyDB (www.taffydb.com) was used for implementing the database.

4. EVALUATION

In order to test this new implementation in a scenario as close to reality as possible, we reused real data arising from a class of students working with the original MiGen system with their teacher in a school. This session involved 15 students who interacted with the eXpresser in the context of a Mathematics lesson for roughly one hour. All occurrences of indicators that were generated during this lesson had been stored in the MiGen database: approximately 1,000 indicator occurrences had been generated by the 15 students working with eXpresser over the one-hour lesson.

For our performance study, we downloaded this set of indicator occurrences into a file and replicated the indicator data as many times as needed, in order to simulate the activity of larger numbers of students interacting with the system in a similar fashion. For each replication of the indicator data, we added a random delay to the timestamp associated with each indicator occurrence (between -2 and +2 seconds for each timestamp), taking care not to overlap with the indicator occurrences before and after. We wrote also a program, to run on a client web brower, that reads this indicator data file and generates the indicator occurrences contained in it at the specified times, just as if this was the MiGen client software running in the web browser. The program also assigns to each student ID a class ID, assuming that there are 30 students in each classroom. From the point of view of the MiGen server software running in the Google cloud, all this activity looks like a set of students all working with eXpresser concurrently, dispersed across their different classrooms.

We evaluated the scalability of the implementation by measuring the average 'round-trip' time taken by an indicator occurrence from the moment that it is generated by the client software until the moment after it is 'pulled' from the MiGen server by the TA tools application and processed so as to update the information being displayed by the TA tools UI (as described in the previous section).

Our first set of simulations was performed by using only one machine to generate all the indicator occurrences and one machine to run the TA tools application. We reached a limit on linear performance at about 25,000 indicator occurrences over one hour (equivalent to the activity of a bit less than 400 concurrent students being monitored by one teacher, which is well above practical classroom sizes), after which it was noticed some of the indicator data began to be lost in transit. Subsequent analysis showed that the reason for this was the web browser running the TA tools application[5]. Browsers allocate a certain amount of memory for user space (regardless of the total amount of memory available on the computer) and we had filled this space up. At this point, the browser started silently discarding some of the indicator data.

In order to overcome this barrier to our scalability study, we decided to use several web browsers in parallel, within the limits of the hardware resources that we could obtain from our institution, which was a computer lab comprising 69 PCs. All these computers were connected to Google's cloud through the university network.

We therefore evaluated our implementation again using these 69 computers. Two thirds of the computers simulated 375 concurrent students each, while the remaining

[5]We trialled with both Firefox and Chrome.

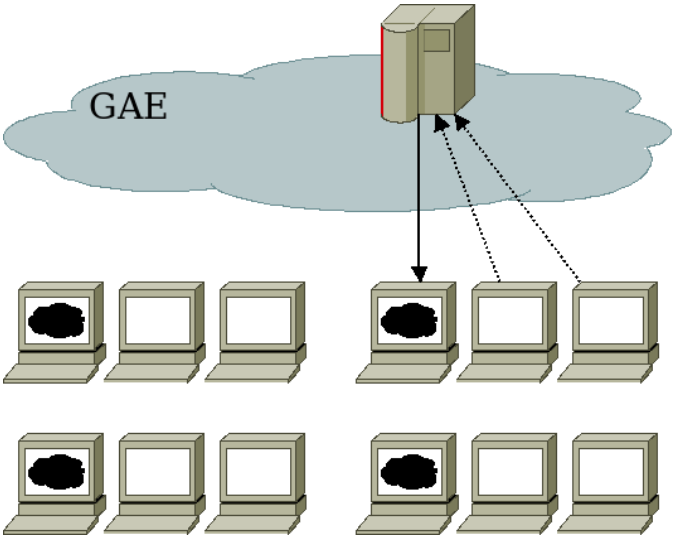

Figure 5: Evaluation setup based on triplets of computers. Two 'student' computers simulate 375 students each, and the third computer runs an instance of the TA tools application, that retrieves indicator occurrences for all of these students. There were 23 of these triplets, 69 PCs in total. The dashed lines represent the computers sending indicators as they occur in the simulation. The solid line represents the TA tools application requesting updates to its local cache of indicator occurrences every five seconds.

computers each ran one instance of the TA tools application, drawing indicator occurrences from the MiGen server for the students being simulated by two corresponding 'student' computers. In other words, there were 23 student–student–teacher triplets of PCs (see Figure 5). As the indicator data being handled by each TA tool application instance was within the limit of 25,000, this now allowed us to stress-test the architecture to its limits.

Results are depicted in Figure 6. The average round-trip time for the indicator occurences in each one of the 23 'contexts' (i.e. triplets of two PCs simulating students and one PC simulating a teacher) was measured. The average delay was under the 5-second upper bound (arising from the 5-second interval between indicator update requests) which is an aceptable delay in the time that it would take for a teacher to be appraised of the current state of a classrom of students. The number of simulated indicator occurrences was approximately 1,150,000, corresponding to the activity of 17,250 students interacting concurrently with the MiGen system over one hour.

5. CONCLUSIONS

We have discussed the challenging role of teachers in supporting students who are working on exploratory learning tasks in the classroom, motivating the need to design tools that support the teacher in being aware of the progress of individual students and of the classroom 'state' as a whole. We have presented a web-based architecture that can support such Teacher Assistance tools, have described an implementation based on standard web and cloud computing

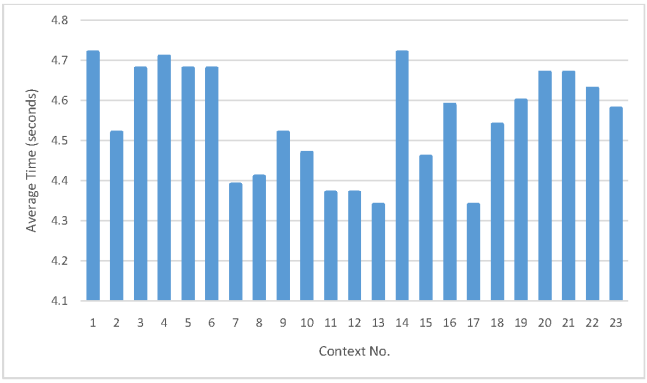

Figure 6: Measurements of average round-trip delay of indicator data updates for the 23 contexts.

technologies, and have demonstrated its scalability to allow concurrent usage by thousands of users, simulating concurrent classes of students and their teachers interacting with an Exploratory Learning Environment. Due to its usage of Google App Engine to store and retrieve the data relating to indicator occurrences, the scalability of our architecture is theoretically limitless.

Although developed and evaluated in the context of the MiGen ELE, our architecture and implementation are generic and could be used for monitoring students' progress in any ELE in which key indicators relating to student-system interactions are detected by the system as students work with the ELE. Our work presented in this paper has demonstrated that it is feasible to provide such ELEs and accompanying Teacher Assistance tools at a national level as software-as-a-service.

Acknowledgements.
This work has been funded by the ESRC/EPSRC TEL MiGen project, with follow-on funding from the Birkbeck School of Business, Economics and Informatics. We thank all members of the project for their help and insights.

6. REFERENCES

[1] O. Amir and K. Gal. Plan recognition and visualization in exploratory learning environments. *ACM Trans. Interactive Intelligent Systems*, 3(3), 2013.

[2] D. Ben-Naim, N. Marcus, and M. Bain. Visualization and analysis of student interactions in an exploratory learning environment. In *1st Int. Workshop on Intelligent Support for Exploratory Environments*, 2008.

[3] C. Cortez, M. Nussbaum, G. Woywood, and R. Aravena. Learning to collaborate by collaborating: a face-to-face collaborative activity for measuring and learning basics about teamwork. *Journal of Computer Assisted Learning*, 25(2):126–142, 2009.

[4] T. Dragon, M. Mavrikis, B. McLaren, A. Harrer, C. Kynigos, R. Wegerif, and Y. Yang. Metafora: A web-based platform for learning to learn together in science and mathematics. *IEEE Trans. on Learning Technologies*, 6(3):197–207, 2013.

[5] R. C. Garcia, A. Pardo, C. D. Kloos, K. Niemann, M. Scheffel, and M. Wolpers. Peeking into the black box: visualising learning activities. *Int. J. on Technology Enhanced Learning*, 4(1/2):99–120, 2012.

[6] V. Gueraud, J.-M. Adam, A. Lejeune, M. Dubois, and N. Mandran. Supervising Distant Simulation-Based Practical Work: Environment and Experimentation. In *EC-TEL*, pages 602–608, 2009.

[7] S. Gutierrez-Santos, E. Geraniou, D. Pearce-Lazard, and A. Poulovassilis. Design of Teacher Assistance Tools in an Exploratory Learning Environment for Algebraic Generalization. *IEEE Trans. on Learning Technologies*, 5(4):366–376, 2012.

[8] S. Gutierrez-Santos, M. Mavrikis, and G. D. Magoulas. A Separation of Concerns for Engineering Intelligent Support for Exploratory Learning Environments. *Journal of Research and Practice in Information Technology*, 44:347–360, 2013.

[9] C. Hoyles, R. Noss, and P. Kent. On the Integration of Digital Technologies into Mathematics Classrooms. *International Journal for Computers in Mathematical Learning*, 9(3):309–326, 2004.

[10] R. Martinez-Maldonado, Y. Dimitriadis, A. Martinez-Mones, J. Kay, and K. Yacef. Capturing and analyzing verbal and physical collaborative learning interactions at an enriched interactive tabletop. *Int. J. CSCL*, 8(4):455–485, 2013.

[11] M. Mavrikis, S. Gutierrez-Santos, E. Geraniou, and R. Noss. Design requirements, student perception indicators and validation metrics for intelligent exploratory learning environments. *Personal and Ubiquitous Computing*, 17:1605–1620, 2013.

[12] M. Mavrikis, S. Gutierrez-Santos, and A. Poulovassilis. Design and evaluation of teacher assistance tools for exploratory learning environments. In *LAK 2016, to appear*.

[13] R. E. Mayer. Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction. *American Psychologist*, 59(1):14–19, 2004.

[14] R. Mazza and V. Dimitrova. CourseVis: A graphical student monitoring tool for supporting instructors in web-based distance courses. *International Journal of Man-Machine Studies*, 65(2):125–139, 2007.

[15] E. Mercier, G. Vourloumi, and S. Higgins. Student interactions and the development of ideas in multi-touch and paper-based collaborative mathematical problem solving. *British Journal of Educational Technology*, 2015.

[16] R. Noss, A. Poulovassilis, E. Geraniou, S. Gutierrez-Santos, C. Hoyles, K. Kahn, G. D. Magoulas, and M. Mavrikis. The design of a system to support exploratory learning of algebraic generalisation. *Computers and Education*, 59(1):63–82, 2012.

[17] A. Pardo. Virtualization increases institutional knowledge of student learning activities. *EDUCAUSE*, 2012.

[18] D. Pearce and A. Poulovassilis. The conceptual and architectural design of a system supporting exploratory learning of mathematics generalisation. In *EC-TEL*, pages 22–36, 2009.

[19] D. Pearce-Lazard, A. Poulovassilis, and E. Geraniou. The design of teacher assistance tools in an exploratory learning environment for mathematics generalisation. In *EC-TEL*, pages 260–275, 2010.

[20] E. Voyiatzaki, P. Polyzos, and N. Avouris. Teacher tools in a networked learning classroom: monitor, view and interpret interaction data. In *6th International Conference on Networked Learning*, 2008.

[21] A. Wichmann, A. Giemza, M. Krauss, and H. U. Hoppe. Effects of awareness support on moderating multiple parallel e-discussions. In *CSCL'09*, pages 646–650, 2009.

[22] V. A. R. Zaldivar, A. Pardo, D. Burgos, and C. D. Kloos. Monitoring student progress using virtual appliances: A case study. *Computers & Education*, 58(4):1058–1067, 2012.