Computers & Education xxx (2011) 1–19

Contents lists available at SciVerse ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

The design of a system to support exploratory learning of algebraic generalisation

Richard Noss [a,*], Alexandra Poulovassilis [b], Eirini Geraniou [a], Sergio Gutierrez-Santos [b], Celia Hoyles [a], Ken Kahn [a], George D. Magoulas [b], Manolis Mavrikis [a]

[a] London Knowledge Lab, Institute of Education, University of London, 23-29 Emerald Street, London WC1N 3QS, UK
[b] London Knowledge Lab, Birkbeck College, University of London, UK

ARTICLE INFO

Article history:
Received 10 May 2011
Received in revised form
5 August 2011
Accepted 20 September 2011

Keywords:
Exploratory learning
Algebraic generalisation
Intelligent support
Teacher assistance
Design

ABSTRACT

This paper charts the design and application of a system to support 11–14 year old students' learning of algebraic generalisation, presenting students with the means to develop their understanding of the meaning of generality, see its power for mathematics and develop algebraic ways of thinking. We focus squarely on design, while taking account of both technical and pedagogical issues and challenges, and provide an account of how we have designed and built a system with a very close fit to our knowledge of students' difficulties with the subject matter. We report the challenges involved in building a system that is both intelligent and exploratory, a learning environment in which both student and teacher are supported without explicit tutoring.

1. Introduction

This paper describes the design of a system built to deal with an intractable problem in mathematics learning, namely the problem of teaching and learning algebra. The key difficulty is that algebra has two faces, which are, for most learners, distinct. The first face is a powerful methodology for thinking about the unknown, making generalisations and testing conjectures, characteristics that are precisely what has led algebra to evolve as the *lingua franca* of mathematical expression. The second face is that of calculation – practicing and understanding the rules of transformation of algebraic expressions. There is no contradiction between these two faces from an expert point of view. But from the point of view of the learner, this second face has come to dominate the first – it is as if one is supposed to learn scales without ever listening to (or composing) any music. The result is that students typically have little idea what they are supposed to learn, or why they might want to learn it. Quite simply, the language of algebra considered as a system of symbols, is unchallenged as a means for experts to express algebraic knowledge (considered as a way of thinking about the general), but is not necessarily well suited for learning.

One solution to this difficulty has been to create alternative representational systems. For example, Roschelle, Kaput, and Stroup (2000: 47–75) designed the *SimCalc* system as a pathway to learning calculus. Our approach, undertaken in the Migen project,[1] has been to design a computational *microworld* that supports 11–14 year-old students by providing them with intelligent feedback during their constructions. By *microworld,* we mean a software system that allows students to explore the structure of objects within the environment, construct and deconstruct their own objects and crucially, explore the mathematical relationships between and within the objects and the representations that make them accessible (Hoyles, 1993; Thompson, 1987). The central idea, borrowed from Seymour Papert, is that the things that matter (from a pedagogical and epistemological point of view), are precisely the things that students have a rationale for, with the power to inspect, manipulate, and investigate.

Our central design aim has been to build a system that affords students a way to access the general while working on the particular; that is, while engaging with a specific case or problem, to develop an awareness of what this would imply for a more general example. We are guided by the principle of *constructionism* (Harel & Papert, 1991), which argues for the pedagogical importance of building things as a way of building mental representations (see also Healy & Kynigos, 2010; Noss & Hoyles, 1996).

* Corresponding author. Tel.: +44 20 7763 2150; fax: +44 20 7763 2138.
E-mail address: r.noss@ioe.ac.uk (R. Noss).

[1] See www.migen.org.

0360-1315/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2011.09.021

We build on a groundswell of software design that has asserted the effectiveness of open, exploratory and expressive tools: see, for examples, (Healy & Kynigos, 2010; Hoyles, 1995; Hoyles & Lagrange, 2010; de Jong & van Joolingen, 1998; Noss & Hoyles, 1996). But, emanating both from our own work and that of others, we have become aware of a critical problem that has yet to be satisfactorily resolved, namely to find a balance between allowing students to express their own ideas, and steering them towards the ideas and activities that address what is to be taught (Kirschner, Sweller, & Clark, 2006; Mayer, 2004; Noss & Hoyles, 1996). This problem is all the more difficult given that the role of the teacher in coordinating or "orchestrating" (Artigue, 2002; Trouche, 2004) the classroom becomes more – rather than less – demanding and complex in the context of students each working with their own computer (and will potentially get worse as the 'computer' is replaced by handheld devices). Our response has been to try as far as possible to build some useful intelligence into our system, which guides the student towards effective interaction and which also supports the teacher through a range of visual tools in conceptualising how her students are progressing and suggesting how they might be grouped for productive discussion. A central computational challenge has been to introduce this support *without* destroying the exploratory and creative potential of microworld interaction. Not only does the system need to monitor the actions and goals of the students (no mean feat given the open nature and lack of structure in exploratory learning tasks) but also it must be careful when making pedagogical interventions not to be overly intrusive or directive. Failure to adhere to this latter point easily results in students doing what they are told – even achieving the learning goals of the task – while avoiding the challenge of building any useful mental structures i.e. learning anything important.

Our focus has been on providing opportunities for learners to develop algebraic "habits of mind" (Cuoco, Goldenburg & Mark, 1996), or "ways of thinking" (Papert, 1972). However, these ways of thinking, despite the obvious structure underlying the domain of mathematics, are difficult to operationalise in a way that can be understood by a machine and this is, of course, crucial if we are to develop systems that offer intelligent support. This means that the standard approach of "Intelligent Tutoring Systems" (ITS), whereby a learner's knowledge is modelled with reference to a particular ontology (i.e. a formal representation of the knowledge domain), is not a realistic proposition.

So rather that offering a formal representation of the knowledge domain, we have designed a system with carefully chosen 'epistemic affordances' which can be used in ways that support students' learning (see McGrenere & Ho, 2000). In the case of our research in the MiGen project, these epistemic affordances support students in the following elements: to perceive structure and find ways to express structural relationships; to identify variants and invariants in patterns; and to recognise and articulate generalisations (Mavrikis, Noss, Hoyles, & Geraniou, in press). Each of these elements of the knowledge domain is an essential characteristic of algebraic thinking, yet none lead *a priori* directly to the identification of a set of intelligent prompts or guides for students. Our approach, therefore, has been to develop an intelligent exploratory environment that has been designed and extended iteratively, attempting to take into account on each iteration any additional or enhanced feedback students appear to need, and to assist them in a relatively unobtrusive but guided manner.

To summarise, we are not building an Intelligent Tutoring System in the traditional sense. An ITS is usually predicated on the assumption that it is possible to derive a mapping between sequences of actions and the learning domain and that this mapping is well-defined in terms of what counts as success or failure. In contrast, in exploratory learning environments the mapping from the students' actions to the trajectory of knowledge developed is ill-defined (Mavrikis, Gutierrez-Santos, Pearce-Lazard, Poulovassilis, & Magoulas, 2010; see also Mitrovic & Weerasinghe, 2009 for a detailed discussion of the ill-defined nature of tasks and of the learning domain). Our recognition of this difficulty has meant that a significant part of our attention has been focussed on supporting teachers by reducing their load and increasing their awareness of the classroom 'state' i.e. gaining an overview of student progress against pre-defined learning goals at both micro and macro levels.

Of course, teachers are well able to judge what constitutes progress and to support their students by appropriate prompts and nudges. Their problem is to cope with 30 children at a time. This paper describes how we have managed to reduce overload by providing computer-based support for some of the students' basic needs, thus allowing the teacher to concentrate on the more complex ones. We also provide teachers with tools that can assist them during the lesson by providing information about students' progress during their constructions.

The paper is structured as follows. In Section 2 we introduce MiGen's microworld – the *eXpresser* – and provide the reader with an overview of its rational and functionalities. In Section 3, we give an overview of the whole MiGen system – its main tools and how these are designed to be used by teachers and students. Section 4 presents the Conceptual Model underlying the system, discussing its purpose and the methodology adopted in developing it. In Section 5, we introduce the system's technical environment, which supports the tools presented in the earlier sections while at the same time meeting necessary requirements in respect of extensibility, performance and installation of the software in schools. In Section 6, we describe students' interaction with eXpresser and illustrate the real-time feedback given to students by the system during a specific construction scenario. Section 7 discusses the evaluation methods we have used and their contribution to the design process. Section 8 presents our concluding remarks and identifies directions for further research.

2. The eXpresser microworld and MiGen activities

At the heart of the MiGen system is the *eXpresser* microworld and its related activities. From an early stage of our research, in collaboration with teachers, teacher educators and consultant experts in the domain of algebraic generalisation in secondary education, we decided that eXpresser would centre around activities that encourage students to identify relationships that underpin figural patterns (Noss et al., 2009). Such activities are standard in the English curriculum. Fig. 1 gives two typical examples.

Our previous work, initial pilot studies on the MiGen project, and a review of the literature suggested that the key intention of such activities should be to help students understand that their algebraic rule for the number of tiles must derive from the *structure* of the pattern, however it is perceived – not some numerical pattern-spotting (12, 19, 26, "guess the next one"). It turns out that the inevitably static (and therefore specific) figure that can be presented on paper is often problematic for students as it lacks a rationale for thinking generally. Consider the technical and challenging use of the word "any" in the examples in Fig. 1. How is the student to know that *her* choice of a particular value is not exactly what is required, and that simply counting will not provide an answer?

So in the eXpresser, students are asked to construct *models* (i.e. structural descriptions of figural patterns), *and* associated algebraic rules. The models can be *animated,* so that the student can see how they – and the rule – changes for different values of the task variable(s).

The task is itself presented to students as an animated challenge in order to discourage students from finding the number of tiles by counting. Rather, students are encouraged to perceive the model structurally and construct it in any way they see fit by identifying

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

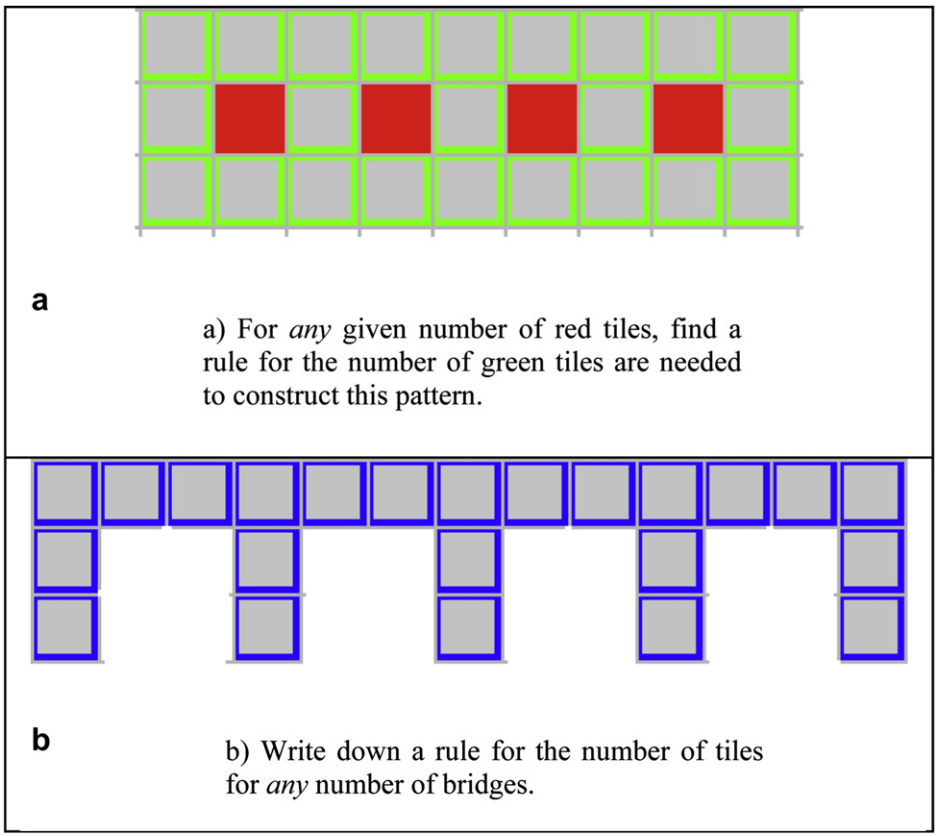

Fig. 1. Examples of curriculum tasks asking students to identify rules underlying a figural pattern.

relationships that they can subsequently express symbolically. The key functionality is that students' constructions are precisely linked to an underlying rule, i.e. *what you build is what you express,* and this rule is clearly aligned to standard algebra.

Fig. 2 shows a snapshot of the eXpresser microworld. The right-hand pane shows the final stage of a task model constructed by a student we will call Jane.[2] (Section 6 discusses in more detail the actions of students like Jane and the feedback and support they receive from the system as they are working with eXpresser). In order to build a general model, students are challenged to make a 'building block' out of unit-square coloured tiles and to repeat it to form a 'pattern', which forms part of their overall model. In Fig. 2, Jane has dragged green tiles on to her canvas to construct a C-shaped building block (labelled (A) in the figure). During their construction, students make use of numbers, some of which they can subsequently 'unlock' to turn them into variables, thus generalising their model. Jane uses a variable she has chosen to call 'no. of reds' (C), to represent the number of red tiles in her model. To create her first pattern, she repeats her C-shaped building block as many times as the value of 'no. of reds', in this case 4. In creating this pattern, Jane has specified that in every repetition the building block should be placed two squares to the right and zero squares down (this part of her construction is not shown in Fig. 2). To complete her model, Jane needs a vertical line of three tiles at the right-hand end of the model, so she creates another pattern made of a single green tile repeated three times (B). Students are guided to translate the structure of their patterns into expressions by colouring their patterns. For example, Jane has specified that her first pattern made of C-shaped building blocks requires 5 times the value of 'no. of reds' green tiles to colour it (C and D).

A key element of eXpresser is that as students are building their constructions in "My Model", another model (referred to as the "Computer's Model") can be seen alongside it (see left-hand pane in Fig. 2). The "Computer's Model" mirrors exactly "My Model" until the student 'unlocks' a number (i.e. creates a variable). At this point, eXpresser randomly changes the value of this variable within the "Computer's Model". This is an example of 'microworld feedback' rather than feedback from the intelligent support of the system (this difference will be illustrated in more detail in Section 6).

The idea of 'locked' and 'unlocked' numbers was introduced into eXpresser so students could specify whether a number should stay the same (remain 'locked' – i.e. a constant) or could change (be 'unlocked' – i.e. a variable). In "My Model" students can edit the values of their unlocked numbers, whereas the "Computer's Model" is read-only and eXpresser chooses random values for 'unlocked' numbers. For example, in the "Computer's Model" in Fig. 2, the current value of the unlocked number 'no of reds' is 2 (E), resulting in a different instance of the model to that currently shown in "My Model". In addition, the "Computer's Model" is coloured by the system only when the student has expressed a correct general colouring rule for their model (F). At any time, the student can click the play button (G) to animate their model and test its generality. The eXpresser animation facility assigns random values to unlocked numbers and displays successive instances of the model.

[2] All students' names have been changed in this paper.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

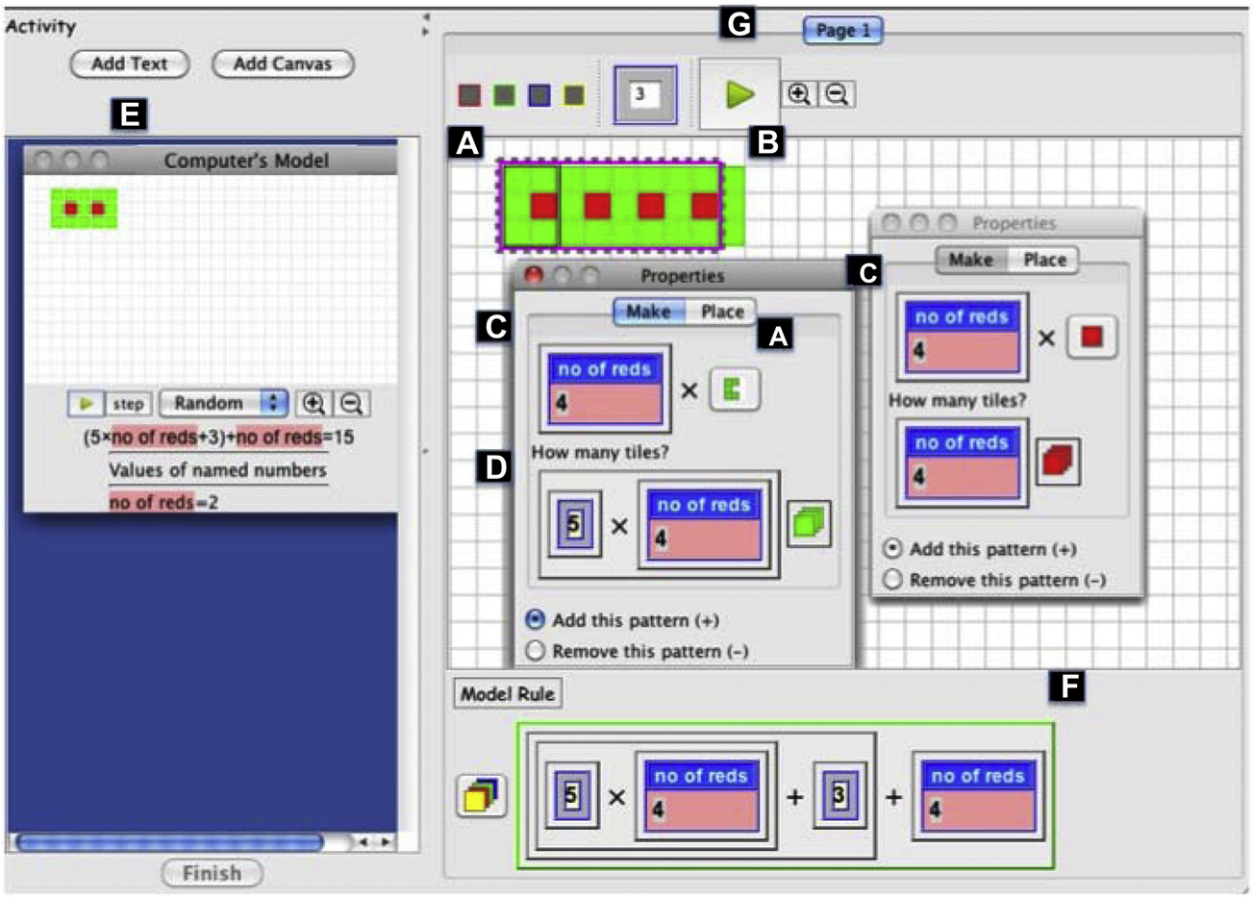

Fig. 2. Constructing the "Footpath" model in eXpresser. Letters highlight the main features: (A) a C-shaped building block made of 5 tiles is repeated to make a pattern; (B) a pattern made out of 1 tile repeated 3 times in the vertical direction to complete the model; (C) the number of repetitions of the building block (A) is 'no of reds'; (D) the number of green (light grey) tiles in this pattern; (E) in the Computer's Model, the system chooses different values for the unlocked number, 'no of reds', and in this case it has chosen the value 2; (F) the student's Model Rule that specifies the total number of tiles in their model; (G) students can click this button to animate their model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Tasks such as the one Jane solved in Fig. 2 are built into activity sequences, which are presented to students through an *Activity Document* generated by MiGen's *Activity & Task Design Tool* (discussed in Section 3). Thus it is through Activity Documents that students are presented with a task sequence, designed by the learning designer or teacher. These include one or more eXpresser models (e.g. to present the task at hand), a list of goals associated with the activity, and questions that students are asked to answer so they reflect on the activity. Fig. 3 illustrates an example of an Activity Document.

If an activity has been designed so as to contain a list of goals (see, for example, the left-hand pane in Fig. 3), students tick off each of the goals when they consider that they have achieved it. When they consider that they have finished a task, they click the 'Done' button and progress on to subsequent tasks (e.g. the reflective questions as shown in the right-hand pane in Fig. 3). As well as students needing to tick off the listed goals, the system itself monitors students' achievement of the goals associated with a task and provides appropriate feedback in the form of suggestions if task goals are not accomplished.

In collaboration with teachers from four secondary schools in England, we have developed activities each of which includes several phases: introduction to a construction task, undertaking the task using eXpresser, reflecting on their constructions, and sharing and discussing their constructions and rules with other students. We have developed introductory activities that aim to familiarise students with eXpresser, and other individual and group activities that target the problem of algebraic generalisation. In all cases, students are asked to construct general models, with the overall goal to develop a quasi-algebraic rule for the total number of tiles, in the form illustrated in (F) in Fig. 2.

After students have constructed their models and derived their rules, they are directed to reflect on both and then to work in groups to justify them. These final group activities are designed to help students consolidate what is general and how generality can be expressed. These collaboration activities are supported by one of the Teacher Assistance tools – the *Grouping Tool* (described in Section 3). During these activities, students are encouraged to make sense of each other's constructions, and persuade each other of their rules' correctness and equivalence (including what is meant, in the context of eXpresser, by "equivalence").

3. MiGen system – functionalities and context

Having seen, in the previous section, what eXpresser tasks involve and how students can undertake activities set them by the teacher, we now focus on the rest of the tools comprising the MiGen system and how they are intended to be used by teachers and students.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

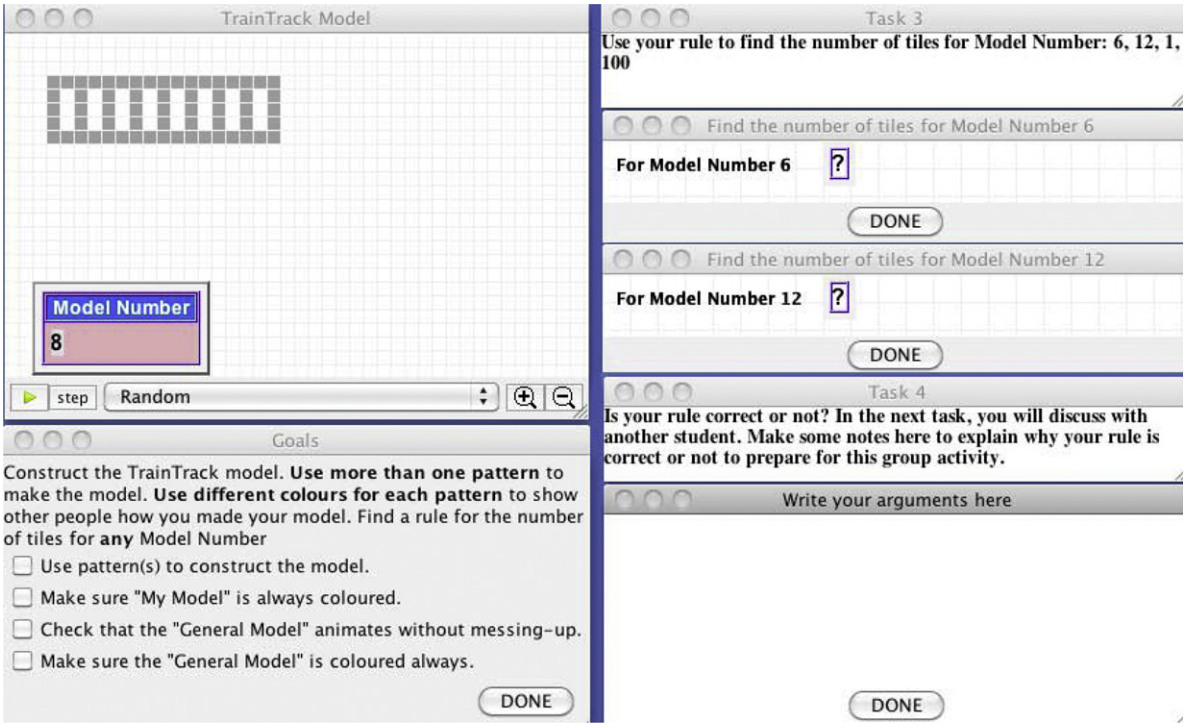

Fig. 3. An Activity Document with the task presentation, a list of goals to be achieved, and reflective questions. The Activity Document is usually presented vertically, and students scroll down as they undertake successive tasks (it is presented here as two side-by-side 'pages' to save space).

In addition to the eXpresser and Activity Documents, the other main student-facing tool is the *eGeneraliser*. This is the system component that is responsible, among other things, for generating feedback messages for students (as will be illustrated in Section 6). The *eGeneraliser* comprises a set of modules that take as their input, information from eXpresser as students are undertaking tasks, as well as information stored in the MiGen database about students, activities and tasks. The eGeneraliser generates the most appropriate form of real-time feedback for students by applying a set of rules that combine information about the student's current construction, the recent history of actions, and feedback messages already displayed to the student (Gutierrez-Santos, Mavrikis, & Magoulas, 2010a; Mavrikis & Gutierrez-Santos, 2010).

As a student interacts with eXpresser, the eGeneraliser also automatically infers a series of *interaction indicators* about the student's construction, selected ones of which can then be notified to the teacher via the Teacher Assistance tools, which we discuss shortly. There are two categories of interaction indicators:

Task-independent (TI) indicators occur when the system detects that specific actions have been undertaken by a student e.g. 'student has placed a tile on the canvas', 'student has made a building block', 'student has unlocked a number'. The detection of TI indicators is not dependent on knowledge specific to the task that the student is currently working on.

Task-dependent (TD) indicators are detected as a result of intelligent reasoning applied by the eGeneraliser to a student's actions. TD indicators are generated from a basis of knowledge about the current task. The inferences generally have a degree of uncertainty. Examples of TD indicators are: 'student has made a plausible building block for this task', 'student has unlocked too many numbers for this task', 'student has coloured their pattern generally', 'student has achieved task goal *n*'.

A variety of intelligent techniques are employed within the eGeneraliser in order to infer the occurrence of TD indicators. For example: an adaptation of case-based reasoning is used to determine if students are employing building blocks that are appropriate for their patterns (Cocea, Gutierrez-Santos, & Magoulas, 2010); rule-based reasoning is used to determine if the student has coloured a pattern in a general or a specific way; and a combination of a sliding window algorithm and string metrics are used to detect rhythm in the actions of the student, which we (and the system) interpret as an emergent recognition of pattern (Gutierrez-Santos, Mavrikis, & Magoulas, 2010b). Rule-based reasoning is also used to update the system's information about students during, and at the end of, their usage of the eXpresser i.e. to update the *Learner Models* (Mavrikis et al., 2010).

The *Activity & Task Design Tool* (see Fig. 4) allows the learning designer (teacher, researcher, etc.) to set-up new activities and tasks for the MiGen system, by selecting from a set of templates provided by the system that determine how the activity will be presented, as well as the questions and instructions for the students. The learning designer can also specify new templates. Subsequently, the designer can select a pre-saved task model for the presentation of the task to the students. To improve the nature of the feedback provided by the system to students, the designer can also provide a set of model constructions that she considers to be possible 'solutions' to the task which students may follow (this helps the system to detect TD indicators such as 'student has made a plausible building block for this task'), as well as expected difficulties that students might face (these will help the system to provide specific support e.g. guide students towards a specific construction strategy).

Other important parts of the description of a task are its goals and learning objectives. Examples of task goals are 'make a building block', 'construct a pattern', 'write an expression for the number of tiles of each colour in your model'. Learning objectives may be epistemic, operationalised or pragmatic. Epistemic objectives capture the MiGen system's objectives as relating to the domain of generalisation, e.g.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

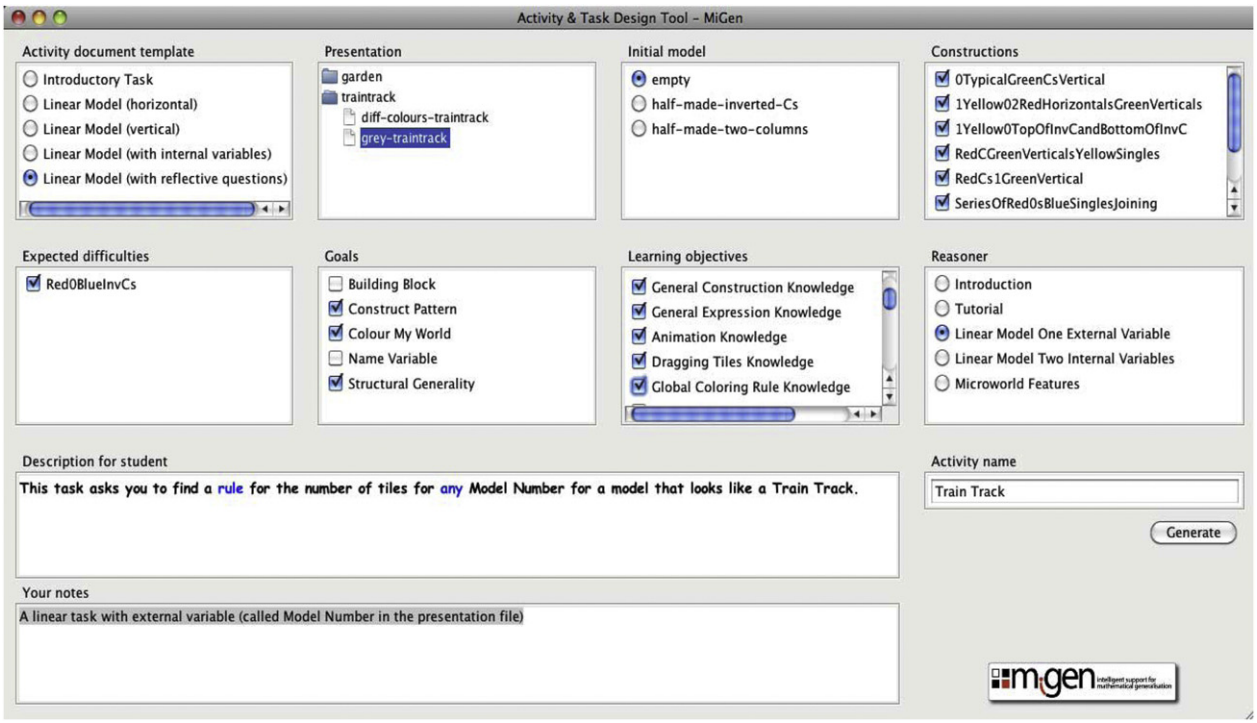

Fig. 4. The activity & task design tool.

'appreciation of the use of variables', 'dynamism as a way to prove generality'. Operationalised objectives contextualise in eXpresser the epistemic objectives e.g. 'can use an unlocked number to create a link between two patterns' contributes to the epistemic objective 'appreciation of the use of variables', while 'use of animation with unlocked numbers' contributes to the epistemic objective 'dynamism as a way to prove generality'. Pragmatic objectives correspond to affordances of eXpresser, largely independently of any epistemic basis, e.g. 'can drag a tile onto the screen', 'can use animation'.

A suite of tools, named the *Teacher Assistance Tools* assist the teacher in monitoring students' activities, informing her real-time interventions in the class, and reviewing students' achievements to assist in future lesson planning. These tools run only on the teacher's computer and provide graphical representations of the occurrence of TI and TD indicators (the Student Tracking tool), the accomplishment of task goals (the Goal Achievements tool), whether students are engaged in the task or not or in need of help (the Classroom Dynamics tool), and the similarities between students' constructions leading to suggested possible pairings for the collaboration activity (the Grouping Tool).

Requirements analysis for these tools has been undertaken with our teacher collaborators, driving their iterative specification and co-design. In this paper, we focus on the *Student Tracking* and *Grouping* tools. Fig. 5 illustrates the Student Tracking Tool that monitors students working with eXpresser. Each column represents the progress of the student named at the top of the column, and each horizontal line indicates the occurrence of an indicator for that student. For example, we see that Heather Sonne[3] began by trying to animate her construction a couple of times, then placed some single tiles on the canvas, and then created a pattern. This last line is shown in red (dark grey), indicating that the pattern was not correct for the task at hand. The colour displayed for an indicator may be: green (mid grey) – meaning positive interaction; red (dark grey) – meaning negative interaction; or yellow (light grey) – meaning interaction that may be either positive or negative depending on the context (for example, Heather's first set of tile placements do not lead to a correct pattern, but her second set of tile placements do). Also, blue is used for feedback messages that are generated for students by the system.

We see that there are also two vertical lines showing in Heather Sonne's column – one labelled "Active" and one labelled "Plausible BuildingBlock". These are 'state' indicators which, unlike the event-based indicators shown as horizontal lines, are monitored continuously by the system. We see that Heather Sonne started as active – her Active line is initially coloured green (mid grey) but then she became inactive – her Active line turns red (dark grey). If the teacher sees this display on her computer, she can intervene and encourage Heather to try to place single tiles in order to construct the pattern. Heather subsequently starts to construct again by placing single tiles. The system detects rhythm in her tile placements – as shown by the occurrence of the event indicator "Feedback-Rhythm detected" – and provides feedback to Heather (with the message "It seems you keep repeating this building block. Why don't you use it to make a pattern?"). The fact that this has happened is displayed in the Student Tracking tool as a blue horizontal indicator occurrence labelled "Feedback: Rhythm Detected". The tool shows that Heather then creates a correct pattern for the task, which results in the Plausible Building Block state indicator turning from red to green.

The *Grouping Tool* is another of the Teacher Assistance tools. This supports the teacher in setting up collaborative activities, by automating the pairing of students based on students' prior constructions. Prior work by the research team suggested that students are likely to engage in productive discussion if they have produced *dissimilar* models and rules. Degree of similarity therefore formed the basis of the pairing logic implemented in the tool. In a normal classroom, identifying appropriate groups by taking into account the construction of each and

[3] All students' names have been changed.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

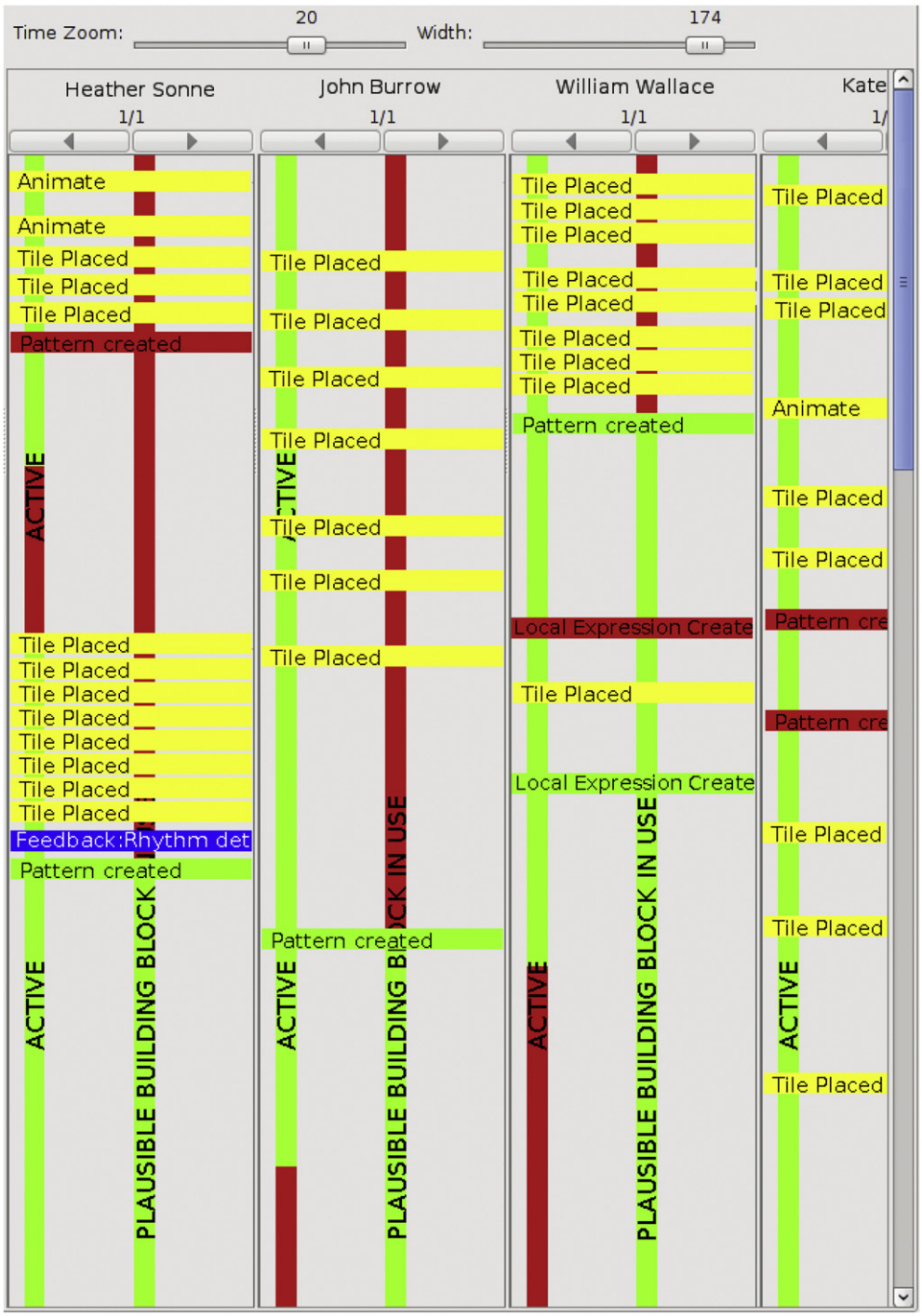

Fig. 5. Student Tracking Tool, showing the occurrence of interaction indicators as students are working on a task. Each student's interactions are displayed within one column. Within each student's column, event-based indicators are shown as horizontal lines and state indicators as vertical lines.

every student is a time-consuming task that is practically impossible to achieve in real-time. The Grouping Tool is designed to semi-automate this process, while leaving the final pairing to the teacher. The pairings generated by the tool are shown visually to the teacher for her to confirm or change (see Fig. 6).

The similarity of pairs of students' models is calculated by the system by comparing different aspects of the students' constructions. These include the building blocks used, the properties of the patterns (e.g. the relative positions of successive building blocks within the pattern), and the 'unlocked' numbers used. Attributes relating to each of these three aspects are represented as vectors of values, V_1, V_2, ….[4] Given two students' models, a similarity metric is applied to the pairs of vectors resulting in a set of numbers s_1, s_2, …, normalised so as to fall

[4] A recent addition is a vector whose value is determined by how the student visualises the model number - i.e. how the student conceives the constant term of the relationship.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

in the range [0,1]. The overall similarity of the two models is given by $\sum w_i s_i$, where the weightings w_i assign relative importance to different aspects of the students' models These weightings have been fine-tuned in consultation with teachers.

The system iterates over all possible combinations of student pairings (and possibly one triplet, in the case of an odd number of students in the class), choosing the one that minimises the overall measure of similarity summed over all pairings. These suggested pairings are presented visually to the teacher: each student is represented by their initials within a circle, and the degree of similarity between pairs of constructions is represented by a small green rectangle, for low similarity (i.e. less than 1/3); medium-sized yellow rectangle, for moderate similarity (between 1/3 and 2/3); or large red rectangle, for high similarity (greater than 2/3). The teacher is able to select students' circles and 'drag' them into different groups to change the system's suggestion if the pairs suggested would not in her judgement, result in productive collaboration.

We refer the reader to Pearce-Lazard, Poulovassilis, and Geraniou (2010) and Gutierrez-Santos, Geraniou, Pearce-Lazard, and Poulovassilis (in press) for further technical details of the design and implementation of the Teacher Assistance tools.

Designing, developing and evaluating the MiGen system has posed both pedagogical and technical challenges, demanding an interdisciplinary approach, and requiring combined expertise from the learning and computing sciences. We have adopted an iterative research and development methodology within the project, comprising successive cycles of:

i. Pedagogical and technical research;
ii. Requirements elicitation, analysis and specification, both within the research team and in collaboration with students, teachers and teacher educators;

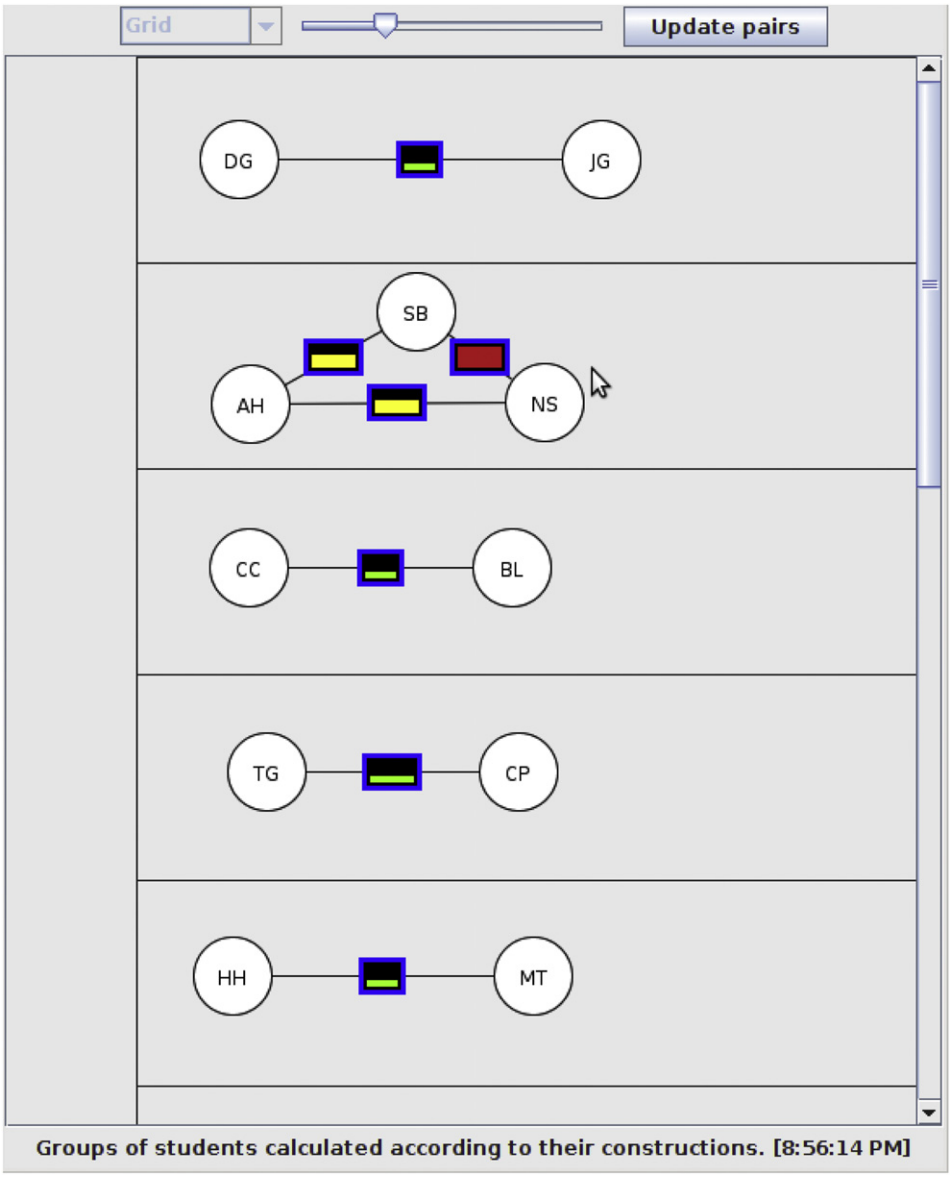

Fig. 6. Grouping Tool User Interface. Each circle represents a student. The groups generated by the system are delineated by horizontal lines. The degree of similarity between pairs of students' constructions is shown as a small green, medium-sized yellow, or large red rectangle, denoting low, medium or high similarity. The teacher can drag students' circles into different groups so as to change the system's suggestion. When the teacher is satisfied with the groupings, she clicks on 'Update Pairs'. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

iii. Development (i.e. design, implementation and testing) of the next system iteration;
iv. Development of activity sequences and tasks to underpin pedagogical and technical evaluation of this iteration;
v. Evaluation studies, both within the research team and with groups of students and their teachers; and
vi. Analysis of evaluation results, elicitation of pedagogical and technical outcomes, and planning of the next iteration cycle.

The early iteration cycles of the project focussed on developing a first version of eXpresser and of the student feedback provided by eGeneraliser (first year of the project). We next turned to development of a first version of the whole system architecture (see Section 5), identification of a first version of the TI and TD indicators, and development of a first version of the Student Tracking tool, while at the same time continuing to enhance eXpresser and eGeneraliser (second year of the project). In the final stages of the project we have focussed on completing the system (developing the remaining Teacher Assistance tools and the Activity & Task Design tool) while at the same time carrying out evaluation studies (see Section 7 below).

4. The MiGen conceptual model

MiGen's *Conceptual Model* was developed as part of steps ii and iii of an iteration cycle that was undertaken early in the second year of the project, to develop a first version of the whole system architecture and the Teacher Assistance tools. A key aim in producing the Conceptual Model was to make explicit the main system entities and the relationships between them. This was a necessary first step in the design of the *MiGen database* which stores the information produced and required by the various tools of the system, and in the design of the overall *MiGen system architecture* which integrates all of the tools described in earlier sections of this paper into a whole system (we discuss the system architecture further in Section 5).

Because, in software engineering, Conceptual Models are at a higher level of abstraction than logical and physical software designs, they are accessible to non-computer scientists and indeed they are often co-designed by broad teams of stakeholders. The MiGen Conceptual Model was co-designed by all members of the project and, in addition to its central aims discussed above, it also served as a successful boundary object (Wenger, 1998) between the various disciplines involved in the project (mathematics education, AI in education, computer science) in gaining an appreciation of their different perspectives, agreeing on common concepts and terminologies, and coming to a common understanding of the main entities and relationships falling within the scope of the system.

Our development of the Conceptual Model began from a series of requirements-elicitation activities with mathematics teachers and teacher educators (examples are reported in Geraniou, Mavrikis, Hoyles, & Noss, 2009; Mavrikis, Geraniou, Noss, & Hoyles, 2008), on the basis of which a first version of the Conceptual Model was produced in early 2009. Subsequent discussions held within the research team resulted in several refinements, modifications and extensions, leading to the final version of the Conceptual Model that we report here, which has formed the basis for the development of the overall MiGen system.

The Conceptual Model comprises a number of overlapping subsections referring to different aspects of the system and we discuss the three major subsections below: (a) Users and Learner Models; (b) Students' Constructions; (c) Tasks, Activity Sequences, Learning Objectives and Landmarks.

4.1. User and learner models

Fig. 7 shows the main entities relating to Users and Learner Models and the relationships between these entities, depicted in the form of a UML Class Diagram. In this diagram, boxes are used to show the entities, and arrows or lines to show the relationships between the entities. An arrow from one entity to another indicates that the former is a subclass of the latter. Lines between pairs of entities are used to depict other types of relationships between entities: at each end of such an edge linking two entities is an indication of the cardinality of that particular end of the relationship and an optional verb phrase.

We see from Fig. 7 that there are three types of user of the MiGen system – Student, Teacher and Researcher; this information is depicted in the diagram by means of three arrows leading from these entities to the User entity. For each task undertaken by a student, the eGeneraliser maintains information within a TaskShortTermModel on the student's ongoing progress through the task: we see from the edge linking the Student and TaskShortTermModel entities in the diagram that a Student 'has' zero or more (0*) TaskShortTermModels, and that a TaskShortTermModel 'belongs to' one Student. TheTaskShortTermModel is subsequently used (again by the eGeneraliser) to derive a longer-term model of the student's strategies and outcomes in relation to this task – the TaskLongTermModel. This, in turn, is used to derive a model of the learner's general understanding of the domain of mathematical generalisation – their DomainLongTermModel. (Thus, a student's overall "Learner Model" comprises their set of TaskShortTermModels, their set of TaskLongTermModels and their DomainLongTermModel.)

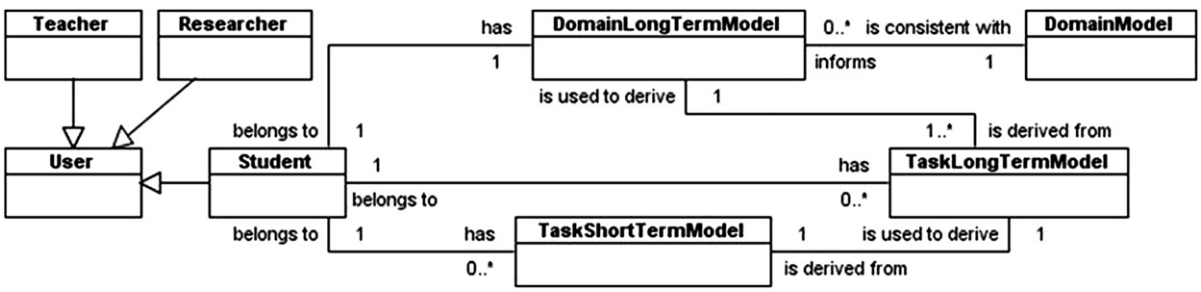

Fig. 7. Entity Relationship Diagram relating to MiGen Users and Learner Models. Boxes denote entities, arrows denote subclass relationships between pairs of entities, and lines other types of relationships between entities.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

Each student's DomainLongTermModel is linked to the overall DomainModel of the MiGen system, which includes concepts such as 'constants', 'variables', 'constructions' and 'expressions', as well as learning objectives mapped from the U.K. Mathematics Curriculum, e.g. 'visualise and draw on grids of different types where a shape will be after a translation', 'understand and use the rules of arithmetic in the context of positive integers', 'explore number relationships and propose a general statement involving numbers'.

4.2. Students' constructions

Fig. 8 shows the main entities involved in students' construction of task models. As a student interacts with eXpresser, the student carries out a sequence of StudentActions. Knowledge of these actions is used by the eGeneraliser to derive and update the student's TaskShortTermModel (which we introduced above). StudentActions also contribute to the student's solution to the task, the TaskStudentSolution. Students' solutions may include TaskExpressions, which are the algebraic rules relating to the student's construction. As they are undertaking the task, the student implicitly transitions through a number of TaskStates, which are automatically detected by the eGeneraliser, e.g. 'student is currently constructing a specific instance of a pattern', 'student is currently constructing a general solution', 'student is creating an algebraic rule'.

4.3. Tasks, activities, learning objectives, landmarks

Fig. 9 shows the main entities and relationships relating to tasks, activities, learning objectives and landmarks in students' constructions. We see that each ExpresserTask is a member of a TaskFamily, which is a conceptual grouping of tasks along various dimensions, such as the number of variables students need to create for the task, and the nature of the algebraic rule. TaskFamilies are operationalised by ActivitySequences, each of which consists of one or more ExpresserTasks that the learner progressively works through (in the current system, each activity sequence is limited to one ExpresserTask, however). Both ExpresserTasks and ActivitySequences have LearningObjectives associated with them (see our earlier discussion of the Activity & Task Design tool). LearningObjectives are expressed in student-oriented language (e.g. within Activity Documents) and each of them related to a corresponding TeachingObjective (e.g. as listed within the Activity & Task Design tool).

As discussed earlier (in our discussion of the Activity & Task Design tool), LearningObjectives may be: system-specific EpistemicObjectives, which capture the MiGen system's objectives as relating to the domain of mathematics generalisation; OperationalisedObjectives, which contextualise EpistemicObjectives in the context of the eXpresser microworld; and PragmaticObjectives, which correspond to affordances of eXpresser independent of any epistemic basis. Both OperationalisedObjectives and PragmaticObjectives can be prerequisites for other OperationalisedObjectives.

As a student constructs a task model within eXpresser, the eGeneraliser may infer and create InferredLandmark entities. Inferred Landmarks include the TD indicators discussed earlier, but may also include additional landmarks relating to higher-level indicators derived from several lower-level TD indicators, for example 'likely to have good understanding of how to use animation to show generality'. The student's actions may also generate landmark entities – ExplicitLandmarks. Again, these include the TI indicators discussed earlier, but may also include additional landmarks relating to higher-level indicators derived from them, for example sequences of indicators. (The current version of the Student Tracking tool supports only TI and TD indicators, and a detailed requirements analysis of higher-level indicators that are useful for teachers is currently under way in the project.) Both these types of landmarks provide evidence of the achievement of OperationalisedObjectives and PragmaticObjectives, as well as for LearnerInconsistencies – these are stumbling blocks such as using more variables than needed for the task.

5. Technical environment

The iterative design of the MiGen system has been derived from the context of how the system is intended to be used within the classroom, the iterative design of the individual tools comprising the system, and the development of the MiGen Conceptual Model. The overall system comprises the *Student software*, the *Teacher software* and the *Server software*. The Student software consists of the eXpresser, Activity Documents and eGeneraliser and runs on each student's computer. The Teacher software consists of the Teacher Assistance Tools and the Activity & Task Design Tool and runs on the teacher's computer. The Server software is responsible for managing the interaction of these user-facing tools with the MiGen database, which stores the information produced and required by the tools.

The MiGen database stores information about TI and TD indicator occurrences detected by the eXpresser and eGeneraliser, so that these can then be presented to the teacher via the Teacher Assistance tools. The MiGen database also stores information about tasks, activities, students' actions and constructions, feedback already given to students, and students' Learner Models. All of these are needed by the eGeneraliser in order to provide personalised, adaptive support to students. The MiGen database is stored centrally on one computer (which

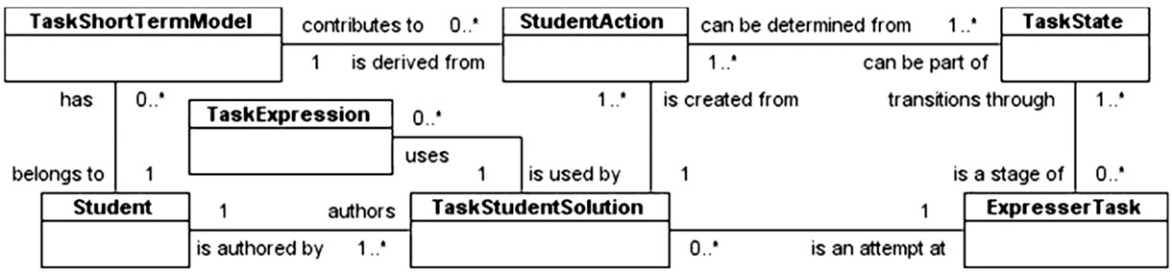

Fig. 8. Entity relationship diagram relating to students' constructions. Boxes denote entities and lines denote relationships between pairs of entities.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

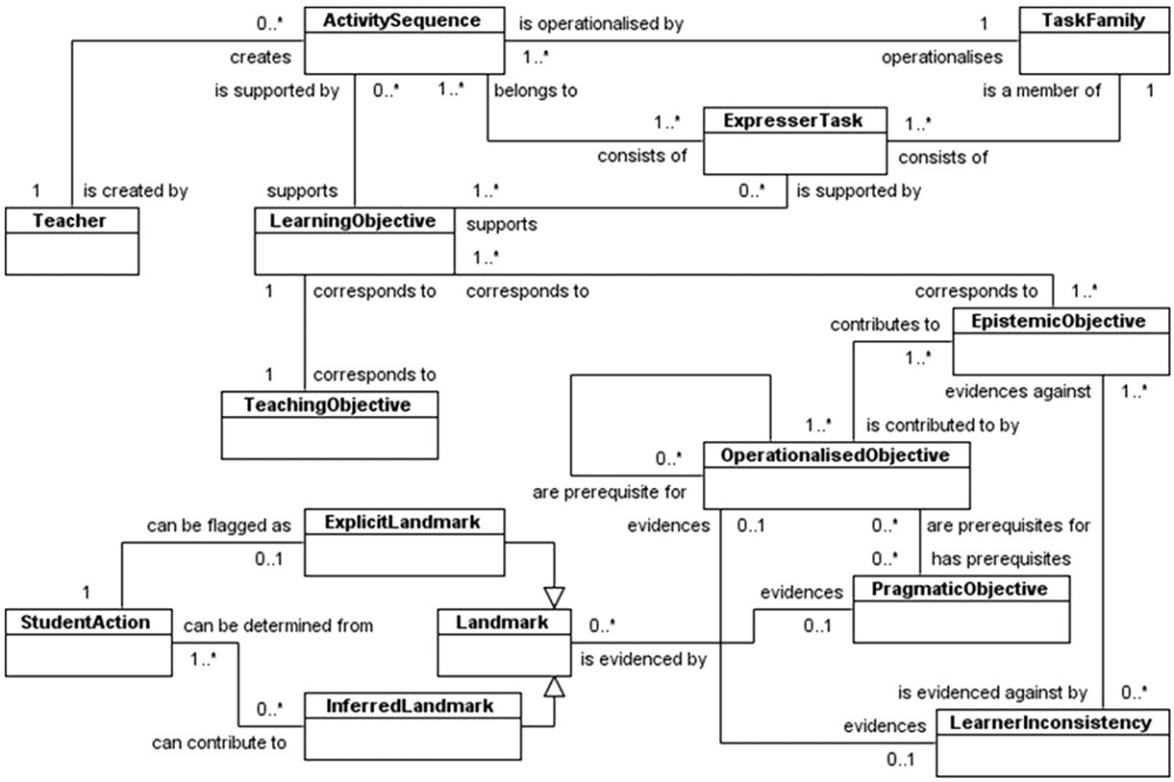

Fig. 9. Entity relationship diagram relating to tasks, activities, learning objectives and landmarks. Boxes denote entities, arrows denote subclass relationships between pairs of entities, and lines other types of relationships between entities.

might be the teacher's computer, or a different machine in the school), and hence the MiGen Server software, which is responsible for managing the interaction of the rest of the system with the database, also runs on that machine. Fig. 10 illustrates the interaction between the students' computers, server computer and teacher's computer in the classroom.

Each of the user-facing tools (i.e. the tools within the Student and Teacher software) 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 underlying MiGen database. The MiGen database stores information relating to the entities of the MiGen Conceptual Model described earlier and it is implemented in JavaDB, a lightweight relational DBMS. The eGeneraliser IL comprises three sub-components: Analysis, Reasoning and Feedback. The Analysis component monitors and analyses the student's actions in eXpresser, and includes a set of modules that implement the intelligent techniques underpinning the detection of TD indicators. The Reasoning layer combines the output of the analysis modules in order to determine when and what feedback should be provided to the student, and also to determine appropriate updates to the Learner Model. The system supports several alternative rule-based 'Reasoners', one for each task 'family' (see Section 4.3), and the applicable one is selected by the teacher/designer through the Activity & Task Design Tool (see Fig. 4). The Feedback component is responsible for generating the actual feedback that will be presented to the student, combining information received from the Reasoning component, knowledge of the student's current construction and recent history of interactions, and the student's degree of completion of the task goals.

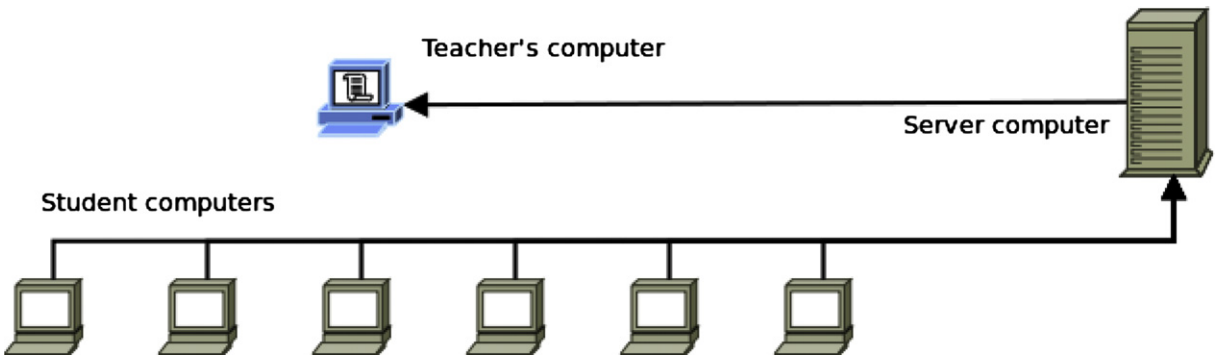

Fig. 10. Classroom set-up of the system. The students' computers send data to the server computer. The teacher's computer receives information from the server computer to present to the teacher.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

As students interact with eXpresser, the eXpresser UI updates the data structures managed by the eXpresser IL. This in turn submits information about students' actions to the MiGen Server, including the occurrence of TI landmarks. The eGeneraliser's IL monitors changes in the eXpresser's IL and from time to time it infers the occurrence of TD landmarks, or updates to Learner Models, which it notifies to the Server software for storage in the MiGen database. From time to time, the eGeneraliser IL may also generate feedback messages appropriate for the student which it displays to the student through the eGeneraliser's UI and also sends to the MiGen Server to be logged. While all this is happening, the Teacher Assistance tools' IL receives information about the occurrence of TI and TD indicators from the Server, which is then presented visually to the teacher by the Teacher Assistance tools' UI components. Fig. 11 illustrates this flow of information between the various components of the system.

The context of use of the MiGen system is very specific: it is intended that the system will be deployed in schools and will be run locally within their own IT infrastructure. Schools' technical administrators are typically reluctant to expose the school systems to unnecessary risks. Also, provision of ongoing technical support, complex server installation, and significant system maintenance would be highly problematic over the longer term. These considerations lead to the need for a lightweight networking architecture. In particular, we have used a Representational State Transfer (REST) approach, due to its simplicity of installation and operation of the system in schools, scalability, extensibility, and ease of performance tuning (Fielding, 2000). We refer readers to (Pearce & Poulovassilis, 2009) for further technical details relating to the implementation of the networking architecture.

6. Using MiGen in the classroom: a scenario

In this section of the paper we aim to give the reader a flavour of what interaction with the system looks and feels like for the student. To do so, we employ a methodological device that has worked well in the past (Noss & Hoyles, 1993) in which we compile *scenarios* comprised of real data from several episodes, to present a 'caricature' of a situation, drawing attention to a range of important elements. The scenario we present below is compiled from our latest study in a suburban school where the teacher ran four classroom sessions using eXpresser in a class of thirty 13-year-old students. Also present during these sessions were two researchers from the MiGen team who provided technical and administrative support to the teacher. We begin with the general context in which data were collected in Section 6.1 before describing the scenario itself in Section 6.2. As this paper is concerned with the *design* of the MiGen system, we refer readers to (Geraniou, Mavrikis, Kahn, Hoyles, Noss, 2009; Mavrikis et al., in press) for detailed pedagogical analysis of the use of eXpresser.

6.1. Classroom activities and context

The teacher started the first lesson by illustrating the main features of eXpresser to students through a simple task called My-Own-Building-Block. This task requires students to build a simple model by repeating any building block they choose, in any direction and any number of times, while making sure that their model can be animated and stays coloured. Fig. 12 shows the task and its goals as presented to students by the system.

Students made several innovative models for the My-Own-Building-Block task, some of which we show in Fig. 13. Students were then set one or two more practice tasks to gain confidence in using eXpresser and familiarise themselves with the system's features.

In the next lesson, students were assigned the TrainTrack activity, illustrated in Fig. 14. This time, the Activity Document presented the TrainTrack task model to students as an animated figure, with the value of the Model Number variable changing alongside the changing figure. The task model was presented in grey (i.e. not coloured) and its description encouraged students to construct this model according to

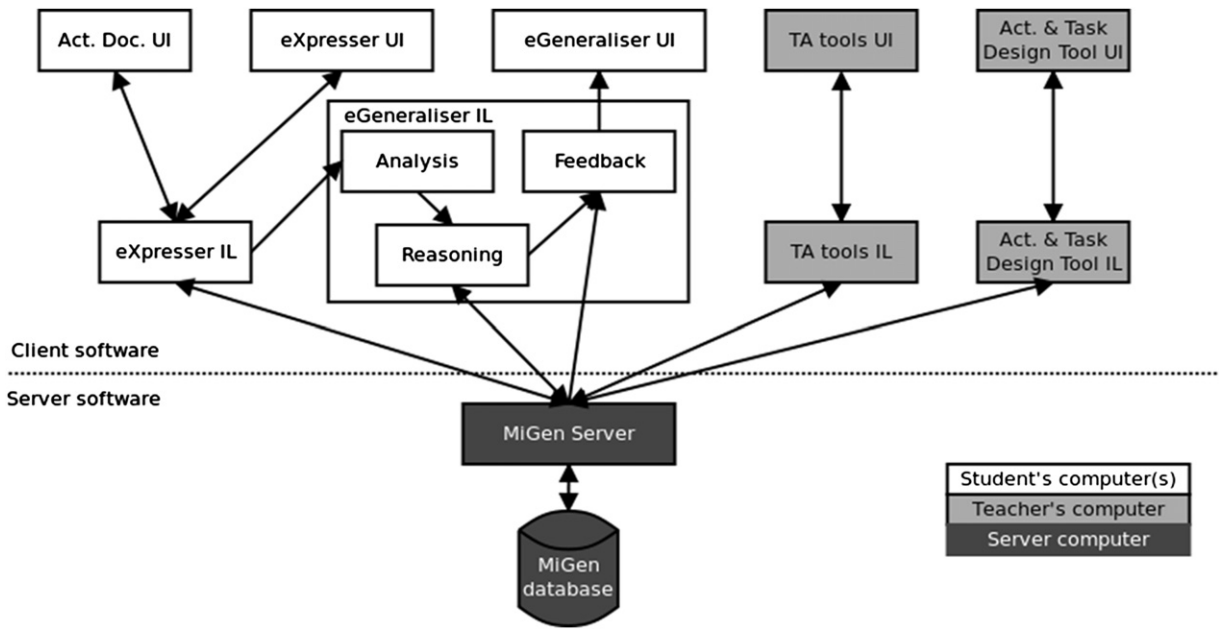

Fig. 11. MiGen System Architecture, showing the components comprising the Student software, Teacher software and Server software, and the information flow between components.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

their perception of the model's structure. The rest of the Activity Document displayed the goals of the activity (similar to the 'Goals' part in Fig. 3). After constructing their model, students were encouraged to reflect on it by answering the following questions listed in the Activity Document, which they can edit to insert their answers: *(1) Use your model to find the number of tiles for Model Numbers 6, 12, 1 and 100. (2) Is your rule correct or not? In the next task, you will discuss this with another student. Make some notes here to explain why your rule is correct or not to prepare for this group activity.*

While students were undertaking the TrainTrack activity, the teacher used the Teacher Assistance Tools to monitor their progress and inform her own interventions in the class. At the end of the TrainTrack activity, students were paired by the system's Grouping Tool and asked to work on a new collaborative activity that involved discussing the correctness and equivalence of their rules. This new activity was also presented to them via an editable Activity Document. This was automatically generated by the system and included the two students' models and rules, and also the following question: *Can you explain why the rules look different but are equivalent? Discuss and write down your explanations.*

6.2. Steve and Laura

Steve and Laura, two of the students taking part in the study, constructed the TrainTrack model in two different ways, as shown in Fig. 15, middle column.

Steve started the task by making a pattern made of 1 red tile repeated vertically 5 times, stating that 5 tiles were required to colour it. He then made a backward C-shaped block of 7 green tiles and repeated it 5 times. He figured out that he required 35 green tiles to colour it. He was happy with his model (it was complete and fully coloured), so he tried to animate it by clicking the Play button. However, the system responded by informing Steve that he hadn't unlocked any numbers, which meant that his model could not be animated. Steve remembered his previous interaction with eXpresser and decided to unlock the number defining the number of repetitions of his green building block. He noticed that the "Computer's Model" changed to a different number of repetitions (as discussed in Section 2, this is designed to help Steve reflect on the notion of a variable). The system, observing Steve's actions, also provided a suggestion: "Your pattern might not be coloured for a different number of building blocks" – as shown in Fig. 16.

Motivated by this suggestion from the system, Steve chose a different value, 6, for the number of repetitions and his model was no longer fully coloured. Steve remembered his previous interactions with the "My-Own-Building-Block" task and asked for help from the system, selecting the question "I am trying to colour the patterns in My Model" – see Fig. 17. He received the response "How many tiles are in the building block? How many building blocks are in the pattern?", designed to help him find a rule that would give him the correct number of green tiles in every case .

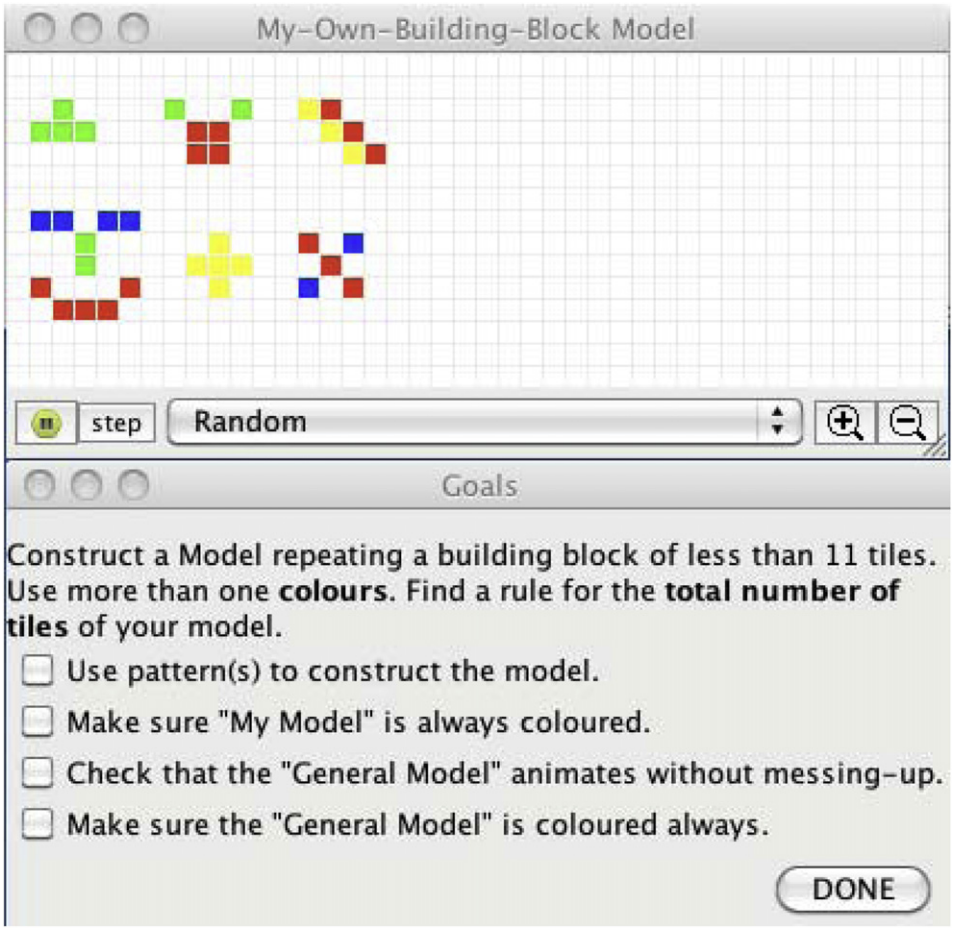

Fig. 12. The Activity Document 'My Own Building Block' co-designed with the teacher.

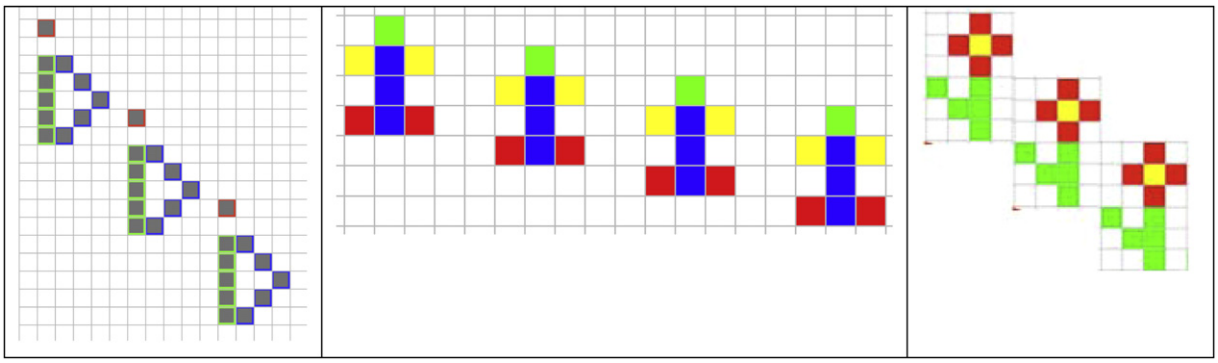

Fig. 13. Examples of students' models in the 'My own building block' task.

Steve realised that he needed to provide an *expression* defining the number of green tiles in terms of the number of repetitions of the building block. In the end, he derived the rule shown in the right-hand column in Fig. 15.

Concurrently with Steve's construction, Laura began the task by making a building block of 5 blue tiles placed vertically and then repeated 4 times to the right. She coloured this pattern by choosing 20 for the number of blue tiles. She continued by making a building block of 2 red tiles placed vertically with a gap of 3 tiles between them and repeated it 3 times to the right. She coloured this pattern by choosing 6 for the number of red tiles. Receiving similar prompts those of Steve, Laura realised that what was needed for her patterns to remain coloured for any number of repetitions of the building blocks. She used the expressions $5 \times n$ (where n is the number of repetitions of the blue building block) and $2 \times m$ (where m is the number of repetitions of the red building block) for the blue and the red patterns respectively. She then decided to test whether her patterns remain coloured for different values of n and m. She unlocked the number of repetitions of both the red and the blue building blocks (n and m), thus in essence creating two variables where really only one is needed (a common difficulty students of her age face, and one that sidesteps the key fact of a relationship between the two variables – precisely what a generalisation would involve). In such cases, the "Computer's model" displays different values for each unlocked number, thus providing 'microworld' feedback designed to demonstrate the model's lack of generality.

At this point the system provided a 'nudge', displaying the message: "The General Model is messed-up" (see Fig. 18). This message, and generally the pedagogical strategy presented here, is designed to support the motivation for generality by challenging students to construct models that are impervious to changing the values of various properties of their construction (Mavrikis & Gutierrez-Santos, 2010). Our use of the term "messed-up" is inspired by a term students employed in previous work with dynamic geometry (Healy, Hoelzl, Hoyles, & Noss, 1994). For the same reason, the system subsequently provokes Laura to check her work by displaying another message: "Would your model be messed-up if you animate it?"

Following this message, Laura animated her model and, realising that it gets messed up, she carried on changing the values of n and m a few more times, choosing every time the specific values she needed for the number of building blocks for her model. The system displayed: "Make sure your patterns are linked, you can use the same unlocked number in several patterns". Laura had not encountered such a situation before and she therefore asked for help from the system by pressing the help button. After several requests for assistance, Laura eventually received the message "The teacher will come to help you as soon as possible". The teacher, who in the meantime was helping other students, returned to examine the Student Tracking Tool display on her computer, saw this request, and realised that her next priority should be in helping Laura. With the help of the teacher, Laura finally realised that she needed to use the *same* unlocked number to link her two patterns. For her red pattern she used m number of repetitions of the building block and $2 \times m$ to colour it. For her blue pattern she used $(m + 1)$

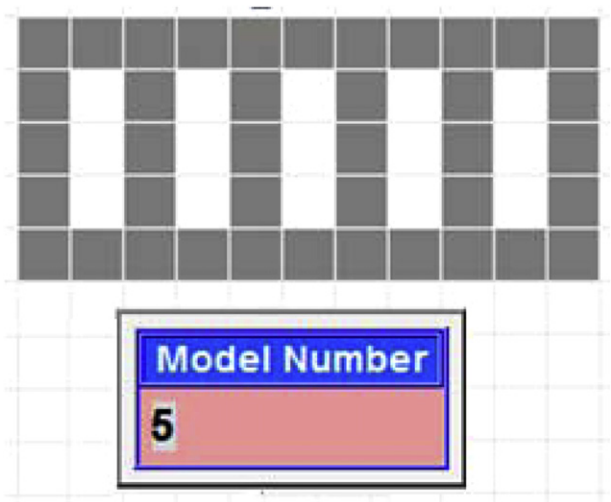

Fig. 14. The TrainTrack model.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

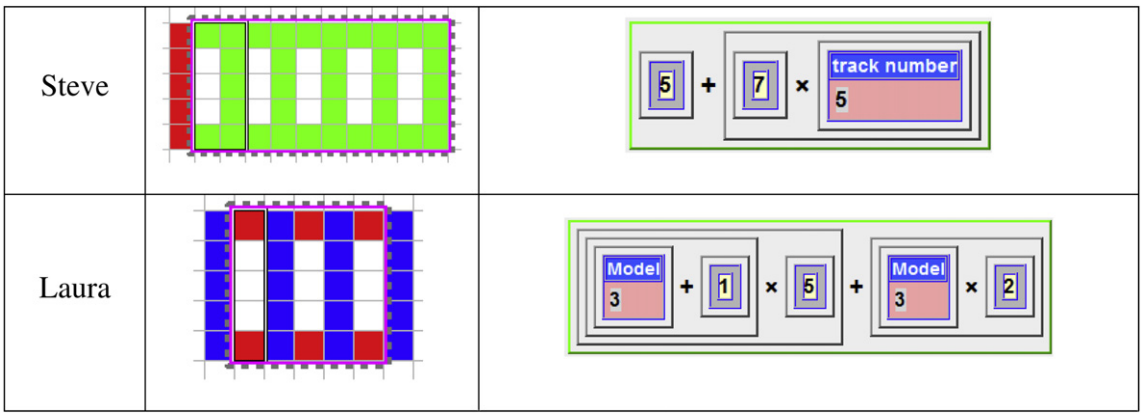

Fig. 15. Steve and Laura's TrainTrack models.

number of repetitions of the building block and $5(m + 1)$ to colour it. Finally, she added these two expressions to derive her overall Model Rule for the total number of tiles – see right-hand column in Fig. 15.

When students finished working on the TrainTrack activity, they submitted their model and rule by clicking the *Done* button in their Activity Document. Back at her computer, the teacher opened the Grouping Tool, which analysed the submitted models and rules and suggested possible groups of students for the collaboration activity to the teacher, based on the degree of dissimilarity of students' constructions. By using the tool to generate the collaboration groups, the teacher was able to ask the students to collaborate immediately after finishing their individual activity, and a new activity document for each group of students was automatically generated by the system. Steve and Laura were grouped together since their approaches were interestingly different and likely to produce constructive discussion. The scenario ended with Steve and Laura becoming reconciled to the idea that their two apparently different rules were, in fact, equivalent and that this could be justified by appealing to the structure of the models they constructed.

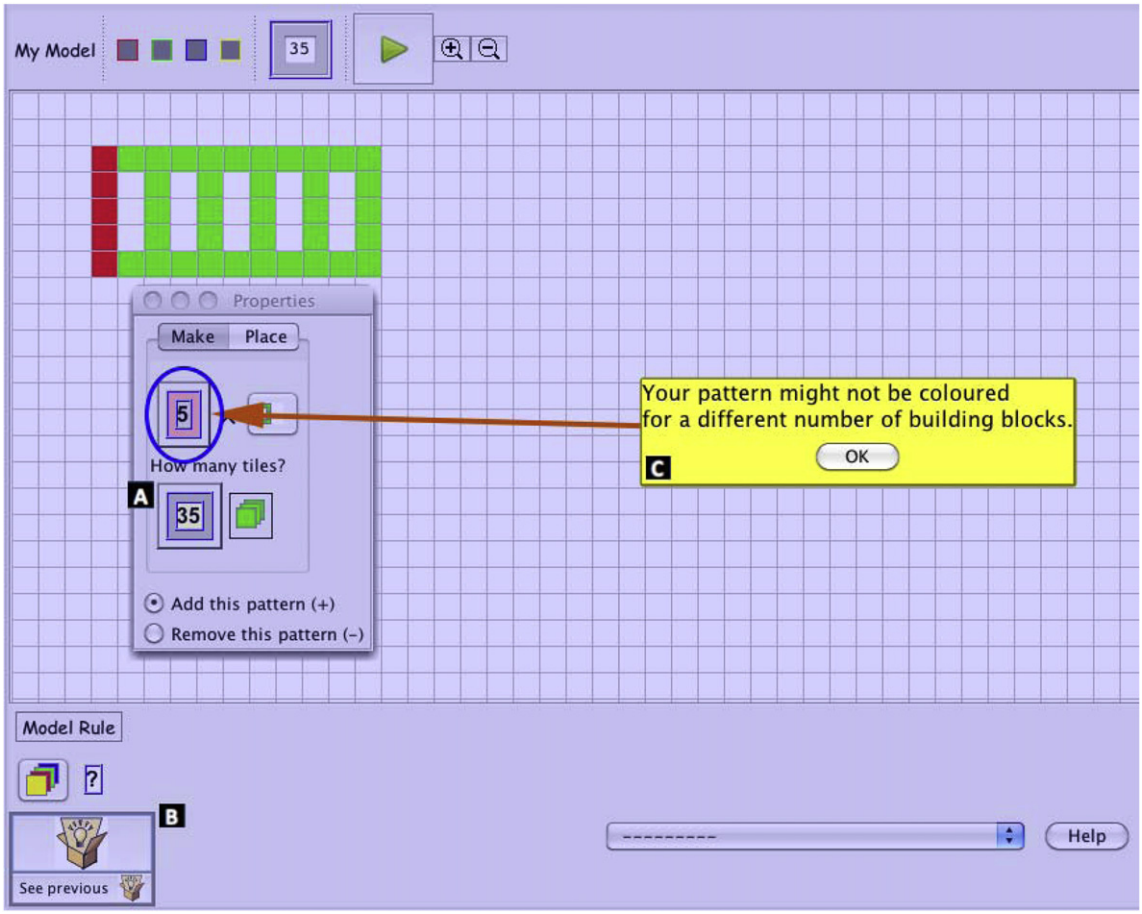

Fig. 16. Steve's pattern is coloured. However he has not provided a general rule for the number of green tiles (A). The system provides a suggestion that Steve accesses by clicking on button B. The suggestion appears, drawing Steve's attention to the unlocked number in his pattern. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

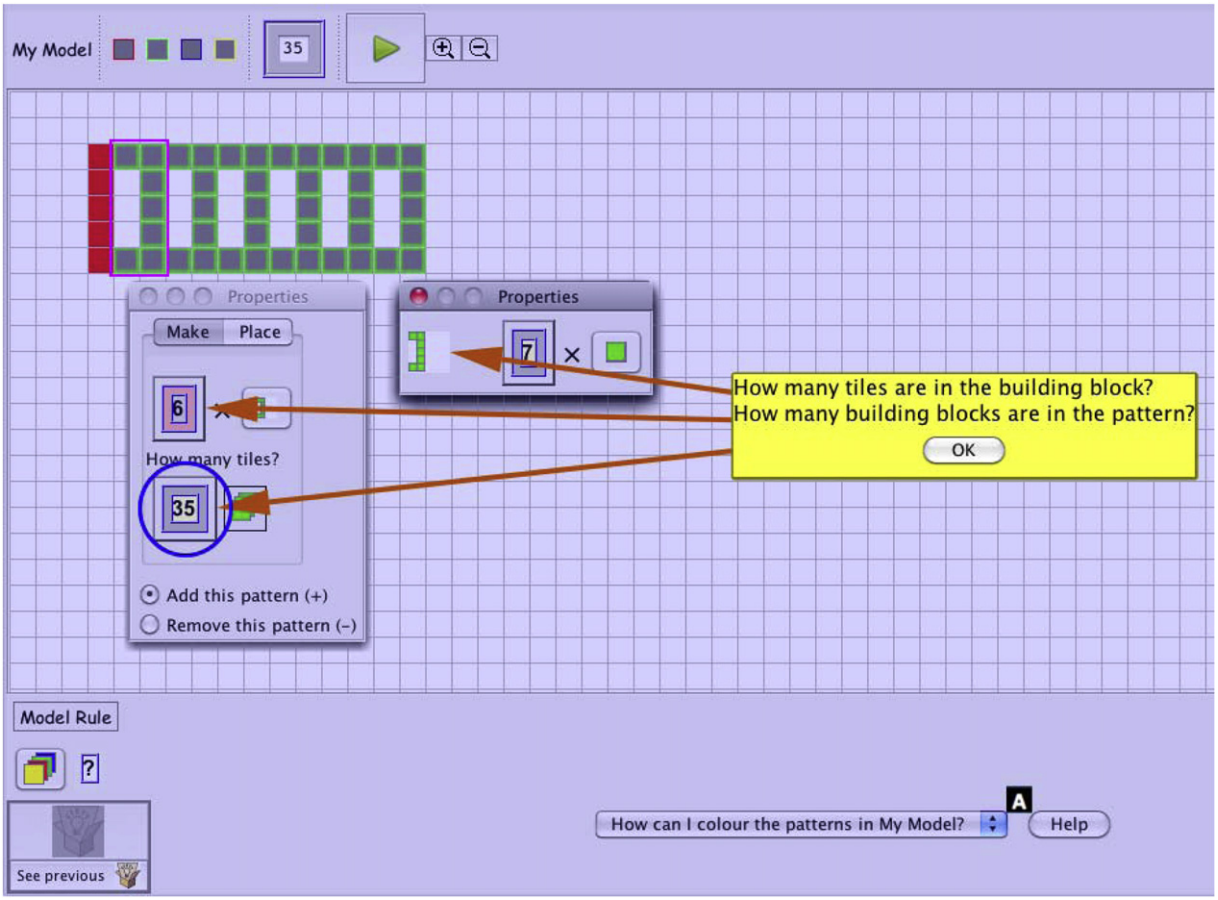

Fig. 17. Steve asks for help using a drop-down menu of possible questions (A). The system, aiming to help Steve find a general rule that would give him the correct number of green tiles in every case, draws his attention to the number of building blocks and the number of tiles in each building block. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7. Evaluation methods

In this Section, we discuss the evaluation methods we have used and their contribution to the design process. Over the course of the project, we have conducted several one-to-one, small-scale, and whole-classroom trials with the MiGen system. We have collected data arising from usage of successive versions of the system, derived from over 300 h of interaction by 11–13 year old learners, mainly from four secondary schools in England. One or more researchers from the project team have been present in these trials acting, as participant-observer. The role of the researcher(s) has been to assist the teacher in ensuring that the technology is functioning, probe students' responses where appropriate, and observe teachers' reactions and methods of incorporating the system into their lessons. The data collected from these trials has been analysed and used to evaluate the effectiveness of the system, and to identify successive improvements and enhancements to both the technical environment (the system's tools) and the pedagogical environment (the activities and tasks).

Early in the project, we elaborated ways in which students' interactions could be supported by the still nascent eGeneraliser. We recognised that what a student knows, and how their learning develops, is mediated by the software and the student's interactions with it and that reciprocally, the software mediation is shaped by students' learning. We therefore designed a series of 'Wizard-of-Oz' studies with small groups of students, in which the system's 'intelligence' is emulated by a hidden human facilitator who supports the student remotely. We engineered didactical situations that allowed us to investigate the impact of the intelligent support without the need to design and implement any software modules in the eGeneraliser before first making sure what was needed (Mavrikis & Gutierrez-Santos, 2010).

The eXpresser provides detailed and meaningful log files that we have been able to analyse in combination with qualitative data from the system trials (Mavrikis & Geraniou, 2011) in order to fine-tune the eGeneraliser. Some of the eGeneraliser's modules make inferences derived from imprecise aspects of students' constructions: for example, the module that determines the degree of similarity of a student's construction to one of the known task solutions. In order to make these modules as effective as possible, we employed a validation process involving pedagogical experts who were asked to evaluate several possible scenarios. Based on the results of this validation, the corresponding modules were fined-tuned.

We have evaluated students' learning primarily through observation on-task, supported by one-to-one task-based interviews with students before and after each activity. Students were questioned in order to probe their understandings and also to obtain their reactions to the technical and pedagogical support provided by the system (which contributed to the iterative design effort). To evaluate students' learning after their interactions with the system, we have designed paper and pencil pre-tasks and post-tasks that students undertake on their own, and these, together with the on-task analysis of students actions and their discourse, are contributing to a description of students' learning trajectories (Baccaglini Frank, Hoyles, Geraniou, Mavirikis, & Noss, in press). Mavrikis et al.,in press presents examples illustrating

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

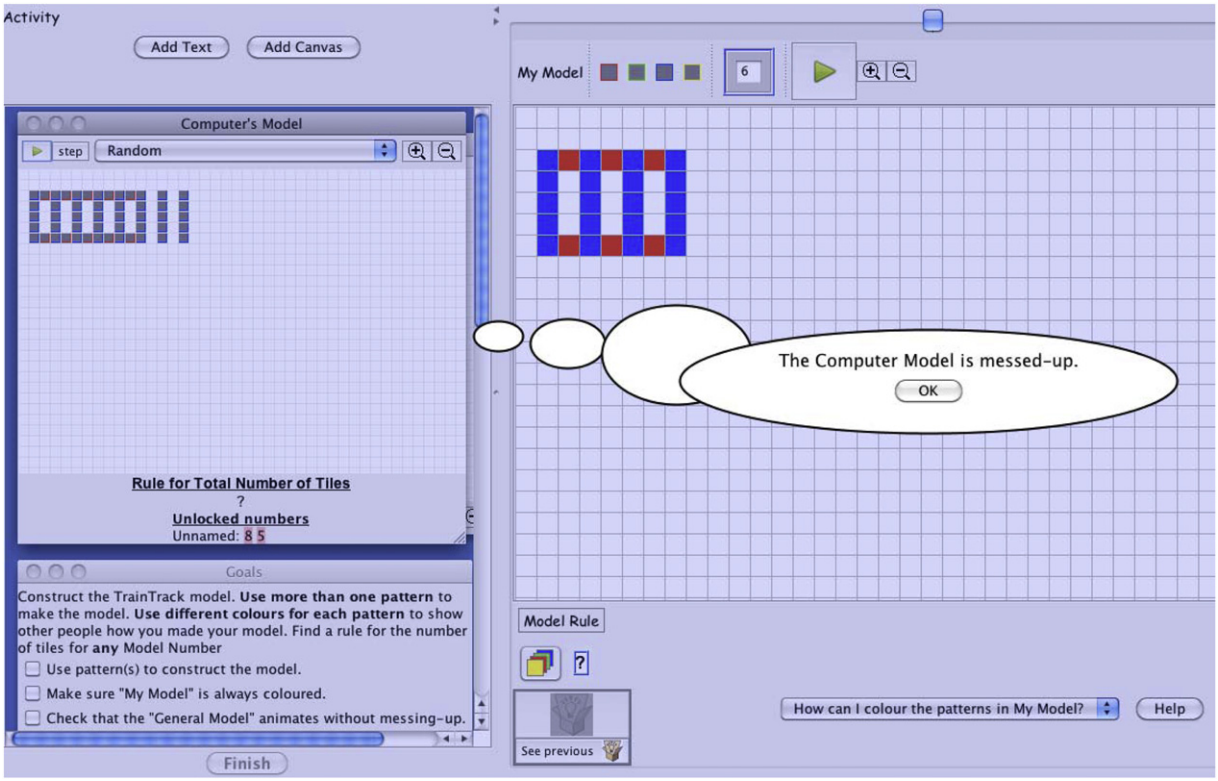

Fig. 18. Laura has coloured her model and all her patterns are general. However, they are not linked and therefore the "Computer's Model" is messed-up. The system draws Laura's attention to this fact. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

how some students successfully came to understand the power of structural reasoning and to recognise how to express relationships explicitly. Of course, we do not expect that a few interactions with eXpresser will enable students to master algebraic generalisation, but we have noted that students after interacting with eXpresser over 3-4 lessons and achieving the learning objectives are able to apply their knowledge to conventional generalisation tasks. We also have some illustrative data that support our conjecture that judicious grouping of students to work collaboratively on different conceptualisations of patterns supports their generalisation techniques in relevant ways, including finding invariants and variants (constants and variables) in models and rules, and expressing relationships using an independent variable to link patterns within models. For more details, see Geraniou, Mavrikis, Hoyles, and Noss (2011).

For formative and summative evaluation of the TA tools, we have held sessions with small focus groups comprising mathematics teachers, teacher educators, and trainee teachers. These sessions have focussed around a core set of usage scenarios targeted by the TA tools, including: awareness of which students need the teacher's immediate help, which students are progressing satisfactorily on the task and which may be in difficulty, identifying common difficulties students are facing in order to provide more explanation to the class, and knowing which students have finished which of the task goals. We have also conducted one-to-one interviews with teachers from two schools after whole-classroom sessions using the MiGen system, again focussing on their use and impressions of the TA tools. The design and results of these evaluations are addressed in a forthcoming paper, Gutierrez-Santos, Mavrikis, Geraniou, & Poulovassilis, in preparation.

8. Conclusions

We have discussed the design of the MiGen system, which aims to support 11–14 year old students and their teachers in the learning of algebraic generalisation. Our focus in this paper has been on questions of design and there are, naturally, a number of detailed methodological challenges that lie beyond the scope of this design-oriented paper. From a design point of view, the challenges we have faced (and continue to face in current work) are a complex mix of technical and pedagogical ones, including:

i. Understanding and identifying the major aspects of the domain of algebraic generalisation, designing the activities and tasks to be undertaken by students, and modelling the range of students' responses;
ii. Identifying the information about tasks, students and students' constructions that needs to be captured by the system in order to underpin the provision of effective feedback;
iii. Designing the feedback to be provided to students, and trialling and deploying intelligent techniques to generate such feedback;
iv. Investigating, trialling and deploying appropriate visualisation techniques for the presentation of useful information to the teacher;
v. Designing and developing an extensible, scalable client-server architecture to support multiple concurrent users in a classroom setting.

A major challenge – arguably, *the* major challenge – has been to design student support in ways that provide students with enough freedom so that they can actively engage in *their* construction task, yet with adequate constraints so as to be able to generate feedback that

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

assists students to achieve *our* goals. Relatedly, the ill-defined nature of the domain continues to pose difficulties for questions of design, as, for example, it is not immediately obvious which aspects of the students' constructions and the current task should be modelled by the system in order to provide effective support; what constitutes a "productive" interaction by the student with the system; and what types of feedback result in more productive student interactions.

A key resolution to these challenges has been the design and implementation of the suite of real-time visualisation tools for the teacher, so that she can follow the progress of her students during the lesson. While there has been previous work on tools to support teachers in an instructionist role, the TA tools represent, to the best of our knowledge, the first work that targets the visualisation of students' mathematical progress through exploratory learning tasks and notifying teachers of students' attainment of specific landmarks (identified from theory and previous research and trialling) as they are undertaking their constructions.

One holy grail for researchers in this and related areas, is to try to pinpoint precisely what kinds of learning are facilitated by this, rather than some other - say, more traditional - pedagogic approach. While the focus of this paper is on design rather than student learning, it is appropriate to add a comment here. We know, from existing research, just how difficult it is for students to generalise mathematically. The literature is replete with examples of students failing to think about the structure of tasks and activities, even when these are designed specifically with that intention. MiGen should make a difference.

There are essentially two possible ways forward. First, we could package all the tools and activities and undertake a set of randomised control trials to compare MiGen "treatment" with a control group. Second, we can analyse what students and teachers *say and do* during pre- and post- interviews and on-task activities, respectively. This we have done, and are continuing to do in, for example, Baccaglini et. al. (in preparation): preliminary analysis shows a substantive evolution in students' discourse when posed, post-MiGen activties, with generalisation problems.

From the point of view of the teachers, the one-to-one interviews with the teachers after each whole-classroom session, point to the effectiveness of the TA tools and the information that can be presented to them by such tools. For the most recent such trial held in June 2011 at a secondary school in England, the TA tools were installed on a tablet PC carried by the teacher as she walked around the classroom. A member of the research team posed a shortlist of questions to the teacher from time to time relating to students' progress e.g. Which students need your help now? Which students are progressing satisfactorily and which may be in difficulty? Which students have achieved task goal number *n*? The teacher was able to use the tools to provide precise answers to these questions, showing a good level of awareness of the status of her class as a whole and of the progress of individual students. A second session was held two days later with the same teacher and class of students, but this time the teacher did not have access to the TA tools, and she reported that she "missed" the information provided by the tools. Details of this and several other evaluation activities relating to the TA tools will be discussed in a forthcoming paper, Gutierrez-Santos et al., in press.

Our teacher collaborators have also identified the need for additional facilities that would allow them to specify and view 'higher-level' derived indicators from the current set of interaction indicators. Teachers have also identified the need to be able to analyse students' progress *after* the lesson, so as to provide more individualised support to students in subsequent lessons. These are both directions for future work.

In conclusion, we note four further challenges. First, our work on intelligent support for students is essentially at a proof-of-concept stage: we have achieved our intention to support a constructionist/exploratory environment, but there is much to do to address the problems of robustness and scalability. Second, we are working on the configurability of our system, exploring the extent to which we can offer teachers the possibility to author their own tasks, without losing the domain-specific support of the intelligent modules. Third, we are redesigning the system to be web-based so as to make it more readily accessible by schools. This will also give us the opportunity to address some user interface issues in eXpresser that are still prototypical in the current system. Finally, we are continuing with further trials and data analysis, to evaluate the impact of the system and the associated activities on supporting the teaching and learning of algebraic ways thinking.

References

Artigue, M. (2002). Learning mathematics in a CAS environment: the genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International Journal of Computers for Mathematical Learning, 7*(3), 245–274.

Baccaglini-Frank, A., Hoyles, C., Geraniou, E., Mavrikis, M. and Noss, R. (in press). Learning algebraic expression in the MiGen system.

Cocea, M., Gutierrez-Santos, S., & Magoulas, G. (2010). Adaptive modelling of users' strategies in exploratory learning using case-based reasoning. In *Proceedings of the 14th international conference on knowledge-based and intelligent information and engineering systems: Part II. KES'10* (pp. 124–134). Berlin, Heidelberg: Springer-Verlag.

Cuoco, A. E., Goldenberg, E. P., & Mark, J. (1996). Habits of mind: an organizing principle for mathematics curriculum. *Journal of Mathematical Behavior, 15*(4), 375–402.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. Irvine: University of California. Ch. Representational State Transfer (REST) (Chapter 5).

Geraniou, E., Mavrikis, M., Hoyles, C., & Noss, R. (2009). A learning environment to support mathematical generalisation in the classroom. In Weigand. (Ed.), *CERME 6, sixth conference of European research in mathematics education*.

Geraniou, E., Mavrikis, M., Hoyles, C., & Noss, R. (2011). Students' justification strategies on the equivalence of quasi-albebraic expressions. In B. Ubuz (Ed.), *Proceedings of the 35th. Conference of the international group for the psychology of mathematics education, Vol. 2* (pp. 393–400), Ankara.

Geraniou, E., Mavrikis, M., Kahn, K., Hoyles, C., & Noss, R. (2009). Developing a microworld to support mathematical generalisation. *International Group for the Psychology of Mathematics Education, 3*, 49–56, PME 33.

Gutierrez-Santos, S., Geraniou, E., Pearce-Lazard, D., & Poulovassilis, A. (in press) Architectural design of teacher assistance tools in an exploratory learning environment for algebraic generalisation (accepted for IEEE Trans. On Learning Technologies).

Gutierrez-Santos, S., Mavrikis, M., Geraniou, E., & Poulovassilis (in preparation), Assisting the teacher in MiGen, an exploratory learning environment for algebraic generalisation.

Gutierrez-Santos, S., Mavrikis, M. & Magoulas, G. (2010a). Layered development and evaluation for intelligent support in exploratory environments: the case of microworlds. In: Proceedings of intelligent tutoring systems (ITS 2010), Vol. 6094 of Springer lecture notes in computer science, 105–114.

Gutierrez-Santos, S., Mavrikis, M. & Magoulas, G. (2010b). Sequence detection for adaptive feedback generation in an exploratory environment for mathematical generalisation. In: Proceedings of the conference on artificial intelligence: Methodology, systems, applications (AIMSA). pp. 181–190.

Harel, I., & Papert, S. (1991). *Constructionism*. Ablex Publishing Corporation.

Healy, L., Hoelzl, R., Hoyles, C., & Noss, R. (1994). Messing up. *Micromath, 10*, 14–17.

Healy, L., & Kynigos, C. (2010). Charting the microworld territory over time: design and construction in mathematics education. *Zentralblatt für Didaktik der Mathematik, 42*(1), 63–76.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021

Hoyles, C. (1993). Microworlds/Schoolworlds: the transformation of an innovation. In C. Keitel, & K. Ruthven (Eds.), *Learning from computers: Mathematics education and technology* (pp. 1–17). Berlin: Springer-Verlag.
Hoyles, C. (1995). Exploratory software, exploratory cultures. In A. Disessa, C. Hoyles, & R. Noss (Eds.), *Computers for exploratory learning* (pp. 199–219).
Hoyles, C., & Lagrange, J. B. (2010). *Mathematics education and technology- Rethinking the terrain*. Springer.
de Jong, T., & van Joolingen, W. R. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research, 68*, 179–201.
Kirschner, P., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential and inquiry-based teaching. *Educational Psychologist, 41*(2), 75–86.
Mavrikis, M., & Geraniou, E. (2011). Using qualitative data analysis software to analyse students' computer-mediated interactions: the case of MiGen and Transana. *International Journal of Social Research Methodology, 14*(3), 245–252.
Mavrikis, M., Geraniou, E., Noss, R., & Hoyles, C. (2008). Revisiting pedagogic strategies for supporting students' learning in mathematical microworlds. In: Proceedings of the international workshop on intelligent support for exploratory environments, at EC-TEL'08.
Mavrikis, M., & Gutierrez-Santos, S. (2010). Not all wizards are from Oz: iterative design of intelligent learning environments by communication capacity tapering. *Computers & Education, 54*(3), 641–651.
Mavrikis, M., Gutierrez-Santos, S., Pearce-Lazard, D., Poulovassilis, A. & Magoulas, G. (2010). Layered learner modelling in ill-defined domains: conceptual model and architecture in MiGen. In: Workshop of intelligent tutoring technologies for ill-defined problems and ill-defined domains. Workshop at the 10th international conference on intelligent tutoring systems (ITS 2010).
Mavrikis, M., Noss, R., Hoyles, C., Geraniou, E. (in press) Sowing the seeds of algebraic generalisation: designing epistemic affordances for an intelligent microworld. In Noss, R. and DiSessa, A. (eds) Special issue on knowledge transformation, design and technology, (Journal of Computer Assisted Learning).
Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction. *American Psychologist, 59*(1), 14–19.
McGrenere, J. & Ho, W. (2000). Affordances: clarifying and evolving a concept. In: Proceedings of graphics interface 2000. pp. 179–186. 8239.
Mitrovic, A., & Weerasinghe, A. (2009). Revisiting ill-Definedness and the consequences for ITSs. In *Proceeding of the 2009 conference on artificial intelligence in education: Building learning systems that care: From knowledge representation to affective modelling* (pp. 375–382). Amsterdam: IOS Press.
Noss, R., & Hoyles, C. (1993). Bob: a suitable case for treatment? *Journal of Curriculum Studies, 25*(3), 201–218.
Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. Kluwer Academic Publishers.
Noss, R., Hoyles, C., Mavrikis, M., Geraniou, E., Gutierrez-Santos, S., & Pearce, D. (2009). Broadening the sense of 'dynamic': a microworld to support students' mathematical generalisation. *ZDM - The International Journal on Mathematics Education, 41*(4), 493–503.
Papert, S. (1972). Teaching children to be mathematicians vs. teaching about mathematics. *International Journal of Mathematics Education and Science Technology, 3*(33), 249–262.
Pearce, D. & Poulovassilis, A. (2009). The conceptual and architectural design of a system supporting exploratory learning of mathematics generalisation. In: Proceedings of the 4th European conference on technology enhanced learning (EC-TEL), Nice. Vol. 5794 of Springer lecture notes in computer science, pp. 22–36.
Pearce-Lazard, D., Poulovassilis, A. & Geraniou, E. (2010). The design of teacher assistance tools in an exploratory learning environment for mathematics generalisation. In: Proceedings of the 5th European conference on technology enhanced learning (EC-TEL), Barcelona. Vol. 6383 of Springer lecture notes in computer science, pp. 260–275.
Roschelle, J., Kaput, J., & Stroup, W. (2000). *SimCalc: Accelerating student engagement with the mathematics of change. Learning the sciences of the 21st century: Research, design, and implementing advanced technology learning environments*, 45–75.
Thompson, P. W. (1987). Mathematical microworlds and intelligent computer-assisted instruction. In *Artificial intelligence and instruction: Applications and methods* (pp. 83–109). Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
Trouche, L. (2004). Managing the complexity of human/machine interactions in computerized learning environments: guiding students' command process through instrumental orchestrations. *International Journal of Computers for Mathematical Learning, 9*(3), 281–307.
Wenger, E. (1998). *Communities of practice: Learning, meaning, and identity*. Cambridge: Cambridge University Press.

Please cite this article in press as: Noss, R., et al., The design of a system to support exploratory learning of algebraic generalisation, *Computers & Education* (2011), doi:10.1016/j.compedu.2011.09.021