The Conceptual and Architectural Design of a System Supporting Exploratory Learning of Mathematics Generalisation

Darren Pearce and Alexandra Poulovassilis

London Knowledge Lab, Birkbeck College
{darrenp,ap}@dcs.bbk.ac.uk

Abstract. The MiGen project is designing and developing an intelligent, exploratory environment to support 11–14-year-old students in their learning of mathematical generalisation. Deployed within the classroom, the system will also provide tools to assist teachers in monitoring students' activities and progress. This paper describes the conceptual and architectural design of the system, and gives a detailed technical explanation of a working proof-of-concept prototype of the architecture, motivating in particular the technologies and approaches chosen to implement the necessary functionality given the context of the project. We also discuss how the prototype will be used as a basis for developing the first full version of the MiGen system, in the context of ongoing knowledge acquisition and analysis within the project's iterative, stakeholder-centred design, development and testing methodology.

Keywords: Exploratory Learning, Mathematics Generalisation, Intelligent Support, Teacher Assistance Tools.

1 Introduction

The use of algebra to express general concepts lies at the heart of mathematics, and the difficulty that algebraic thinking poses for children has been widely studied (e.g. [1,2]). The MiGen project is aiming to co-design, develop and evaluate, with teachers and teacher educators, a pedagogical and technical environment for improving 11–14-year-old students' learning of mathematical generalisation.[1] The idea of 'seeing the general through the particular' is a powerful way of introducing students to generalisation [3]. In the MiGen project, we are adopting a constructivist approach, allowing students to create and manipulate patterns and expressions and to perceive the relationships between them. Our aim is to support students' construction of their own models through exploration while at the same time fostering their progressive building of knowledge [4].

There has been little previous work on supporting students in a constructivist context (e.g. [5]) and specifically in microworlds (e.g. [6,7]). Conversely, it has been argued that considerable guidance is required to ensure learning with constructivist instruction [8]. The nature of our learning environment requires that feedback be provided to students by the system during their modeling process, as well as at the end of it. Since

[1] See http://www.migen.org/ for details of the project's aims and background.

U. Cress, V. Dimitrova, and M. Specht (Eds.): EC-TEL 2009, LNCS 5794, pp. 22–36, 2009.

students will be undertaking exploratory rather than structured tasks, teachers need to be assisted in monitoring students' activities and progress by appropriate visualisation and notification tools. These will assist the teacher in focussing her attention across the class and will inform her own interventions to support students in reflecting on their constructions, on the system's feedback, in setting and working towards goals, and in communicating and working with others [9].

Designing, developing and evaluating a system such as this poses both pedagogical and technical challenges, demanding an interdisciplinary approach. The MiGen project team comprises researchers from the mathematics education, AI in Education, and computer science disciplines. The pedagogical and technical challenges that we are currently addressing in the project are: (i) understanding and modeling the mathematics generalisation domain, the tasks to be undertaken by learners and the learners themselves; (ii) identifying the information about learners' constructions that needs to be captured in order to underpin the provision of effective feedback by the system to learners and to the teacher; (iii) designing the feedback for the learner, and trialling and deploying appropriate intelligent techniques to generate such feedback; (iv) similarly, investigating, trialling and deploying appropriate visualisation techniques for the presentation of feedback to the teacher; and (v) designing and developing an extensible, scalable client-server architecture which will support multiple concurrent users (students and teachers) in a classroom setting, and will readily allow the incremental development and evaluation of the various components of our system during the course of the MiGen project, and beyond.

This paper focuses primarily on (v), including also some aspects of (i) and (iv). We refer the reader to [10,11] for discussion of possible techniques underpinning (iii), and to [9] for discussions of (i) and (ii). This paper is structured as follows. Section 2 sets the scene by giving an overview of the envisaged context and functionalities of the MiGen system. Section 3 discusses the Conceptual Model of the system and the methodology we have adopted in developing this. Section 4 presents our overall system architecture, which aims to encompass the system functionalities presented in Section 2 while also meeting the necessary requirements in respect of extensibility, performance and easy installation within schools. Section 5 gives technical details of an architectural proof-of-concept prototype, motivating the technologies and approaches chosen to implement the necessary functionality. We also discuss how this prototype will be used in the coming months as the basis for developing the first full version of the MiGen system. We give our concluding remarks in Section 6.

2 The MiGen System Context and Functionalities

The MiGen system will be deployed in classrooms within schools. During a lesson, students will be working on mathematics generalisation problems as selected by their teacher and presented to them by the system. Students may be working (individually or in groups) on different variants of the same problem or on different problems. During the lesson, the teacher may wish to view real-time representations of the students' activities and progress. At other times, teachers may also wish to access historical information about their students' activities and progress as maintained by the system.

The MiGen system will comprise a number of tools:

(i) An **Activity Design and Activity Management Tool.** This supports the visual design, storage and enactment of activity sequences targeting maths generalisation. These activity sequences typically include phases such as: introduction to a task; undertaking a task (using the eXpresser tool — see below); reflecting on a task; and discussing constructions with other students (using the Discussion Tool — see below). We are considering using the LAMS system[2] to provide this kind of functionality (excluding the construction and discussion functionality provided by the eXpresser and Discussion Tool). Currently, activity sequences are being designed by members of the research team. In the longer term, we expect that teachers will reorganise existing activities and create new ones.

(ii) The **eXpresser.** This is a mathematical microworld [12,13] supporting students in undertaking maths generalisation tasks which have been specificied using the Task Design Tool (see below). The current version of the eXpresser supports individual construction; in the longer-term we anticipate also supporting design of maths generalisation tasks to be undertaken collaboratively by a number of students. Students' actions undertaken within the eXpresser are stored by the system within an eXpresser Session Log.

In the eXpresser, students are asked to construct 'generalised patterns' using a set of facilities that have been co-designed with teachers and teacher educators. Fig. 1a and Fig. 1b show two instances of a pattern that students may be asked to construct. The eXpresser supports them not only in constructing patterns but also in deriving expressions, such as the number of green tiles required (light grey) given the number of blue tiles (dark grey). For example, if a student has constructed their pattern using a series of 'C' shapes (as illustrated in Fig. 1c), they may derive an expression such as $5b + 3$ for the number of green tiles, where b is the number of blue tiles. Other possible constructions that students may follow are shown in the remaining diagrams of Fig. 1 which also show that the form of the resulting expressions can vary widely (though all of them, if correct, will be equivalent to $5b + 3$ of course). The constructions in Figs. 1c-g are *general* in that they make use of the pattern construction facilities of the eXpresser. As a result, changing the number of blue tiles will lead to the student's pattern changing appropriately. For example, setting $b = 4$ will give the pattern of Fig. 1a and setting $b = 5$ that of Fig. 1b. We refer the reader to [14,15] for further details of the eXpresser.

(iii) A **Discussion Tool.** This will allow students to view and discuss other students' constructions.

(iv) A **Task Design Tool.** This will support the specification of maths generalisation tasks to be undertaken using the eXpresser. In the first instance, this will be a research-team-oriented tool. In the longer term, it will evolve into a teacher-facing tool which allows the designer/teacher to associate learning objectives with a given task and to construct possible solutions for the task so as to enable comparison against students' actual constructions by the intelligent components of the system, e.g. to detect which construction approach a student is likely to be following, or whether they are possibly off-task. We refer the reader to [10,11] for discussion of a case-based reasoning approach to undertaking such analyses.

[2] See `http://www.lamsinternational.com/` for more details.

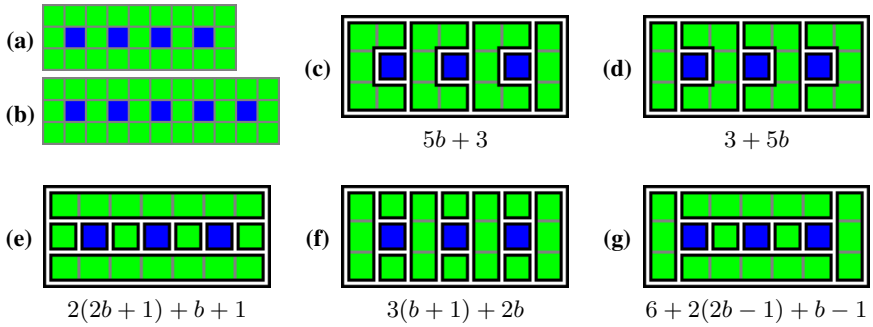

Fig. 1. (a)–(b) Instances of an example pattern and **(c)–(g)** several possible general constructions where each expression specifies the number of green tiles in terms of b, the number of blue tiles

(v) The **eGeneraliser.** This is a suite of intelligent components which take as their input information from the eXpresser as students are undertaking tasks, as well as information stored in the MiGen database relating to students (their learner model) and to tasks. These components generate real-time feedback for students and update their learner model attributes as appropriate. The eGeneraliser will provide different types of feedback, e.g. prompts to help students start a task, identify errors in their solutions, generalise their solutions, seek help from the teacher, etc. (See [9,11] for further discussion). Information about what feedback has been given to each student will be stored within the MiGen database (in the Feedback Log). This will be used by the Teacher Assistance Tools (see below) to allow the teacher to see how the system is interacting with her students (as well as for analysis purposes by the research team).

(vi) The **Teacher Assistance Tools.** This is a suite of tools aiming to assist the teacher in monitoring students' activities and progress, and intervening with additional support for students as she decides appropriate. In the first instance, these tools comprise a **Classroom Dynamics (CD)** Tool and a **Student Tracking (ST)** Tool. The CD tool will provide a visual overview of students' locations in the classroom and their progress with respect to ongoing tasks so that the teacher can view different facets of their activities and use this to inform her choice of interventions within the classroom. The teacher will be able to select from a range of information to visualise, some of which will be derived from the eXpresser Session Log (e.g. which students have started to use variables in their construction, which have submitted their construction, which have created an algebraic expression for their construction, etc) and some from the Feedback Log and Learner Model data generated by the eGeneraliser (e.g. which construction approaches students are using, which students may be off-task, which students have achieved a particular "landmark" in their construction such as moving from a specific to a general solution). Similarly, the ST tool will provide teachers with information about individual students' progress with respect to a given task so that the teacher can follow their progress on this task over time, as well as intervene as appropriate within the classroom to set new goals for an individual student. Some of this information will be derivable from the eXpresser Session Log (e.g. whether the student has completed all aspects of a task) and some from the Feedback Log and Learner Model (e.g. what the system has inferred about the student's constructions, what feedback the system has given to the student).

3 The MiGen Conceptual Model

As stated in Section 1, the pedagogical and technical challenges posed in designing, developing and evaluating a system such as MiGen necessitates an interdisciplinary approach. An iterative methodology has been adopted on the project, comprising successive cycles of:

i. pedagogical and technical research;
ii. requirements elicitation within the domains of mathematics generalisation and intelligent support;
iii. requirements analysis and specification in collaboration with students, teachers and teacher educators;
iv. development of activity sequences and tasks to underpin evaluation of the technical deliverables;
v. development (design, implementation and testing) of technical deliverables;
vi. evaluation of the technical and pedagogical environment, both within the research team and with groups of students and their teachers; and
vii. analysis of evaluation results, elicitation of pedagogical and technical outcomes, and planning of the next cycle.

In this section, we discuss the MiGen Conceptual Model, which falls under item (iii) above. In Section 4 we present the MiGen System Architecture, which falls under item (v). Producing a Conceptual Model for the MiGen system has aimed to make explicit the key system entities and the relationships between them as a necessary first step in the development of the system. The process started from the earlier requirements elicitation activities, reported in [4,9,11], on the basis of which a first version of the Conceptual Model was produced by the authors. Discussions were then held with the rest of the research team, resulting in several refinements, modifications and extensions, and leading to the second version that we report on here. The overall Conceptual Model (CM) comprises four ovelapping subsections referring to the following aspects (breaking down the overall CM into a number of subsections facilitates its description and inclusion of diagrams within an A4 document): (a) Users and Learner Models; (b) Students' Constructions; (c) Tasks, Activity Sequences, Learning Objectives and Landmarks; and (d) Task Solutions. For reasons of space, we present just the first three of these here and refer the reader to [16] for the full CM (including entity attributes).

3.1 Users and Learner Models

Fig. 2 shows the entities relating users and learner models and the relationships between them. There are three types of user of the MiGen system: student, teacher and researcher (this relationship is indicated in the diagram by means of single-headed arrows 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 (by way of explanation of the notation, at each end of an edge linking two entities is an indication of the cardinality of that particular end of the relationship and an optional verb phrase; so a Student 'has' zero or more TaskShortTermModels, while a TaskShortTermModel 'belongs to' one Student). The

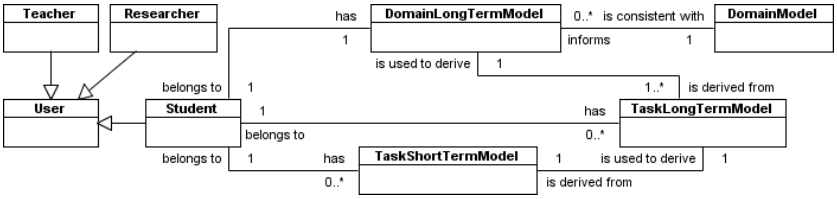

Fig. 2. Entity Relationship Diagram for users and learner models

TaskShortTermModel is subsequently used (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). This links with the overall DomainModel of the MiGen system (there is only one instance of this entity) which includes concepts such as 'constants', 'variables', 'constructions' and 'expressions'. (Thus, a student's "learner model" consists of their TaskShortTermModels, TaskLongTermModels and DomainLongTermModel).

3.2 Students' Constructions

Fig. 3 shows the entities involved in students' construction of patterns. In interacting with the eXpresser, the student generates a sequence of StudentActions, arising from creating patterns and changing the attributes of patterns. This data is used by the eGeneraliser to derive and update the student's TaskShortTermModel (introduced above), and StudentActions contribute to the student's overall solution to the task (the TaskStudentSolution). Solutions may include TaskExpressions, which are part of the task specification and require the learner to answer algebraic questions about their constructions. During the task, the learner implicitly transitions through a number of TaskStates as they create and manipulate their construction. Examples of task states are: 'student is currently constructing a specific instance of the pattern', 'student is currently constructing a general solution' and 'student is creating an algebraic expression'.

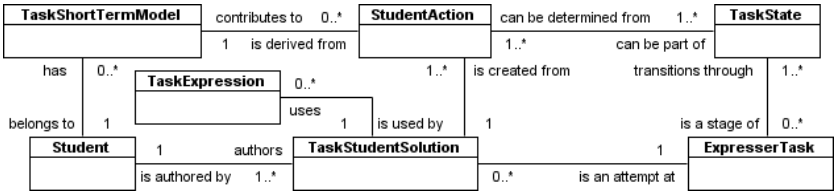

Fig. 3. Entity Relationship Diagram for students' constructions

3.3 Tasks, Activities, Learning Objectives, Landmarks

Fig. 4 shows the entities and relationships relating to these aspects of the Conceptual Model. 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 addressed by ActivitySequences, each of which consists of a number of ExpresserTasks that the learner progressively works through. Both ExpresserTasks and ActivitySequences have LearningObjectives associated with them. LearningObjectives are expressed in student-oriented language, and each of them is related to a corresponding TeachingObjective.

LearningObjectives of tasks and activity sequences correspond to system-specified EpistemicObjectives which capture objectives in the domain of mathematical generalisation, e.g. 'appreciation of the use of variables'. OperationalisedObjectives (also system-specified) serve to contextualise EpistemicObjectives in the context of the MiGen system. For example, an OperationalisedObjective such as 'can use a variable to create a link between two patterns' would contribute to the EpistemicObjective 'appreciation of the use of variables'. PragmaticObjectives correspond to affordances of the eXpresser that are independent of any epistemic basis, e.g. 'can drag a tile onto the screen'. Both OperationalisedObjectives and PragmaticObjectives can be pre-requisites for OperationalisedObjectives.

As a student constructs a solution, the eGeneraliser may infer and create InferredLandmark entities, which indicate events such as the student starting to construct generally rather than specifically. The student's actions may also generate landmark entities — ExplicitLandmarks — as a result of completing or highlighting points in their

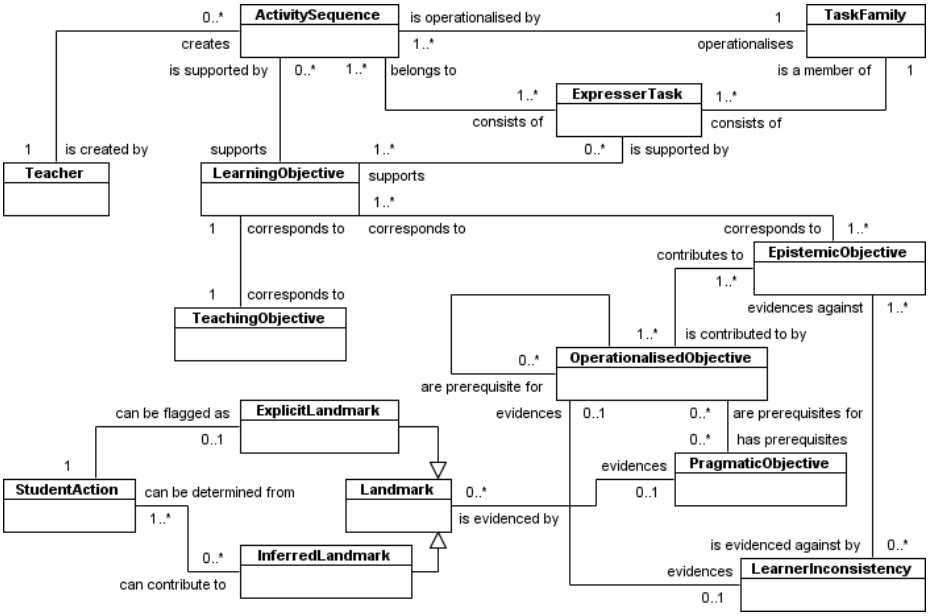

Fig. 4. Entity Relationship Diagram for Tasks, Activities, Learning Objectives, Landmarks

construction process that they may wish to reflect on later, or to discuss with their peers or teacher. Both these types of landmarks provide evidence for OperationalisedObjectives and PragmaticObjectives, as well as for LearnerInconsistencies — these are system-specified stumbling blocks, e.g. using more variables than needed.

4 The MiGen System Architecture

The MiGen system architecture has been iteratively designed from ongoing identification and specification of the tools that will comprise the system and of the context of how the system will be used within the classroom (as discussed in Section 2).

Fig. 5 gives a high-level component diagram illustrating the various components of our architecture and the information flow between them. Each of the user-facing tools features uniformly within the architecture, consisting of a user interface (UI) component (within the overall MiGen User Interface), an information layer for managing the client-side data structures necessary to support the UI component, and a server-side server component. The tools' server-side components all make use of the MiGen Data Server which provides access to the underlying MiGen Database through its 'data accessor' Application Programming Interface (API). The data accessor is an abstraction over the data stored in the MiGen database. It provides facilities so that other components can obtain results from specific queries which are updated in real-time. This abstracts away low-level details of the underlying data storage from the other components. The MiGen database includes the eXpresser Session Log, Feedback Log, Learner Model data referred to earlier and, more generally, information relating to all the entities of the Conceptual Model presented in Section 3.

As they are interacting with the eXpresser and the Discussion Tool, the eXpresser Information Layer posts students' actions to the MiGen Data Server via the eXpresser

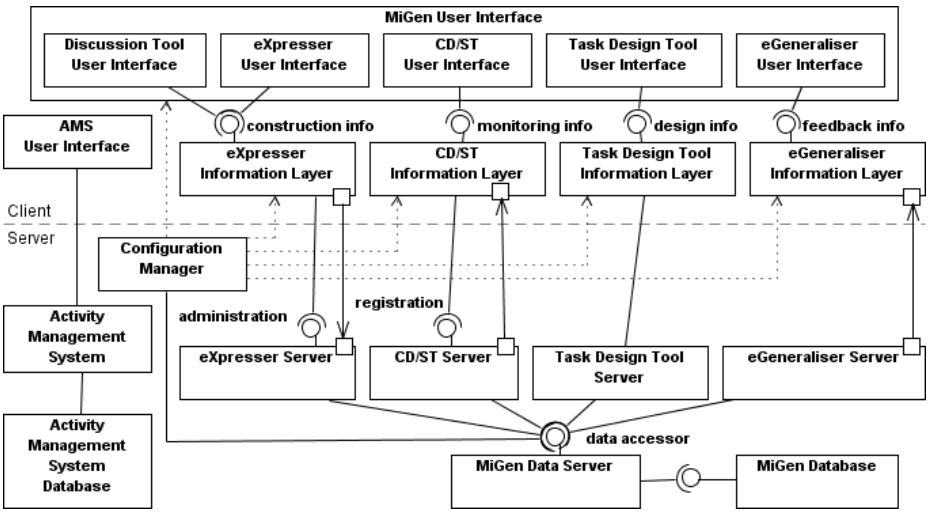

Fig. 5. High-level system architecture

Server. The CD/ST and eGeneraliser tools work similarly except that in their case it is the MiGen Data Server that posts information to their information layers, via their servers, and this information is then presented to the teacher or student by their UI components.

The Configuration Manager is responsible for the application of system-wide settings, as specified in configuration profiles stored within the MiGen Database. Issues relating to the integration (if any) of a pre-existing Activity Management System with the MiGen architecture are being discussed in ongoing work so we have shown no explicit communication links between them. We refer the reader to [16] for a more detailed architectural breakdown of the high-level components shown in Fig. 5.

5 Architectural Proof-of-Concept

We now present a proof-of-concept implementation of the above architecture which focuses on the communication between the client-side components and the server-side components. It constitutes a significant first step towards building the infrastructure to support the first full version of the MiGen system in the coming months.

The context of usage of the system is very specific: the system will be deployed in schools. During the lifetime of the project, we anticipate running a single server instance within our university's IT infrastructure, facilitating data gathering and iterative development. Once the project has finished, however, we intend to provide the system to schools so that they can run it locally within their own IT infrastructure since sustainability is an important project aim. These considerations constrain the server-client architecture in a number of ways. Perhaps the most serious constraint is that there is often not sufficient technical administrative support within schools. Indeed many system administrators are teachers with a technical background who have volunteered for the role. Regardless, they are typically very reluctant to expose the school systems to any more risks than are strictly necessary. This is a potential issue both when running the server centrally since it would not be feasible to open up a new 'MiGen Server' port within school firewalls, as well as when the schools later run the servers locally since provision of technical support, complex server installation and/or maintenance would be highly problematic. These considerations serve to preclude possibilities such as Glassfish and other J2EE-based application servers, as well as servlet containers such as Tomcat and JBoss; the networking architecture needs to be lightweight using a port that is already open.

In view of these constraints, a simple solution would therefore be a lightweight server-client technology that works over HTTP. An obvious candidate is RPC over XML or SOAP. However, a compelling alternative is to build an architecture based on Representational State Transfer (REST) [17]. REST has many advantages over RPC-based approaches: it is (more) cacheable and scalable and is also easier to performance-tune and debug. Within a RESTful architecture, all data resources within the system can be made available at their own unique URL. In the context of our system, this has the additional advantage that students would be able to make a construction publicly available at its URL. This would then be accessible not only within the MiGen system but also using standard web browsers (with suitable content negotiation) thus allowing,

for example, the student's parents to view the work. Given these advantages, we have chosen to use Restlet, a lightweight Java REST framework.[3]

5.1 Implementation Overview

The aim of our proof-of-concept implementation is to design and develop sufficient server-client infrastructure so as to demonstrate that it can fulfil the requirements of the architecture of Fig. 5. At this stage therefore, the intention is not to have a complete architecture, and the following discussion of the proof-of-concept implementation focuses on the core of the necessary functionality.

Our prototype implements a simple server-client architecture where each client can post textual messages to the server. Fig. 6 provides a UML class diagram showing the salient server-side classes which utilise Restlet. The server-side functionality of the prototype is managed by an instance of the ServerApplication class which consists of a set of AbstractRestlets. An AbstractRestlet provides some convenience functionality for calling the appropriate abstract Java method given the HTTP request method. For the purpose of our prototype, only GET and POST request methods are handled, calling getRepresentation() and addRepresentation() respectively. The ClientsRestlet manages the data resource located at `/clients`, which lists the clients using the system. The MessagesRestlet manages the data resource located at `/clients/{id}/messages`, where `id` corresponds to the (numeric) id of an existing client; this lists the current set of messages for the given client (GET) and allows posting of new messages (POST). Finally, the RegisterRestlet provides a mechanism for each client to register their own server location to facilitate server-initiated client communication (see below). The DataModel class manages and provides access to the underlying data of the prototype.

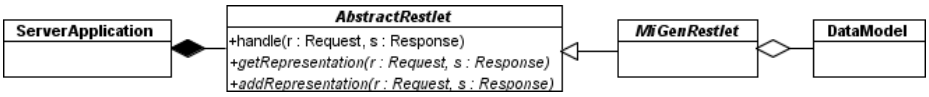

Fig. 6. UML class diagram of the core server-side architecture

The UML class diagram for the core client-side architecture is shown in Fig. 7. For the purposes of this prototype, all data resources consist of a list of items, such as identifiers or messages. The pivotal generic class ResourceList[4] is responsible for providing access to the data resources located at one of the URLs given above, ensuring that the local copy is synchronised with the version on the server. Instances of this class are created via the ResourceListManager which provides a cache of the ResourceLists used by the client. Each ResourceList makes use of a parser for the content of its URL, which is a ListRepresentationParser since all data resources are list-based in this proof-of-concept implementation; also, all representations are text-based and the only parsers defined are based on one item per line of text.

[3] See `http://www.restlet.org/`

[4] The generic parameter is indicated in the annotation on the top right corner of the class box.

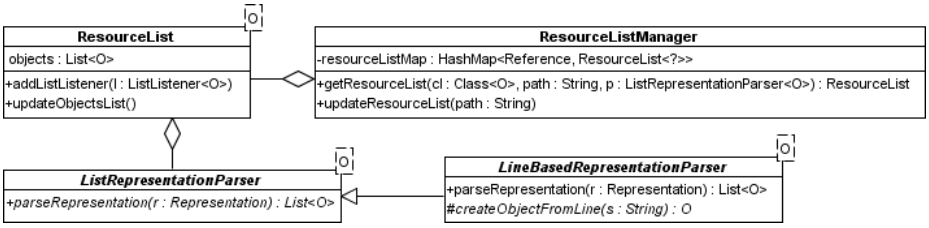

Fig. 7. UML class diagram of the core client-side architecture

Fig. 8 presents a UML sequence diagram illustrating the interaction between the client-side and server-side classes. When the MessagesRestlet is first created by the server, it attaches itself as a listener to events coming from the DataModel (message 1). When the client initialises, it creates its ClientServer 2 which registers with the RegisterRestlet 3 which in turn stores the location of the client server for the particular client 4. The client then creates ResourceListManager 5 and uses it to obtain a ResourceList 6, 7. This concludes the initialisation messages; the remaining messages within the diagram illustrate interaction that happens at various points in response to client activity. In this example, this activity consists of the client posting a new message 8 to the MessagesRestlet. The restlet in turn posts the message to the underlying DataModel 9 which notifies the MessageRestlet's listener 10. In response to this event, the MessagesRestlet asks the DataModel to notify all connected clients that its representation has now changed 11. The client's server is subsequently notified 12 and requests an update to the appropriate ResourceList using the ResourceListManager 13, 14. At this stage, the client's ResourceList has not been updated; all that has happened is that the server has notified the ResourceList that it must re-check with the server to find out the latest state of its data resource. It therefore asks the MessagesRestlet for the latest representation of its data resource 15. The MessagesRestlet contacts the DataModel 16 in order to obtain the updated data resource content 17. This is returned as plain text which is subsequently parsed by the ResourceList 18. The ResourceList then determines which elements have been added or deleted from the list, synchronising with the version of the list held on the server. In this example, a single element has been added so the client is notified appropriately 19.

This demonstrates how a client is able to post new information and receive updates to the existing information in real time. The example shows how a client posting new information only sees it once the server has been notified (which in turn then notifies other clients as illustrated). In the full system, infrastructure will be further developed so that the client will be able to function to some extent without assuming the presence of the server. This will allow local manipulation of a ResourceList directly, synchronising in the background with the server as and when it is available.

It is important to note here that — consistent with RESTful principles — it is the client *not* the server that is responsible for calculating the difference between the server data resource content and the content of its own local copy; the server maintains no client state. This necessarily places a higher computational load on the client but leads to a scalable architecture: computational power increases with the number of clients. As

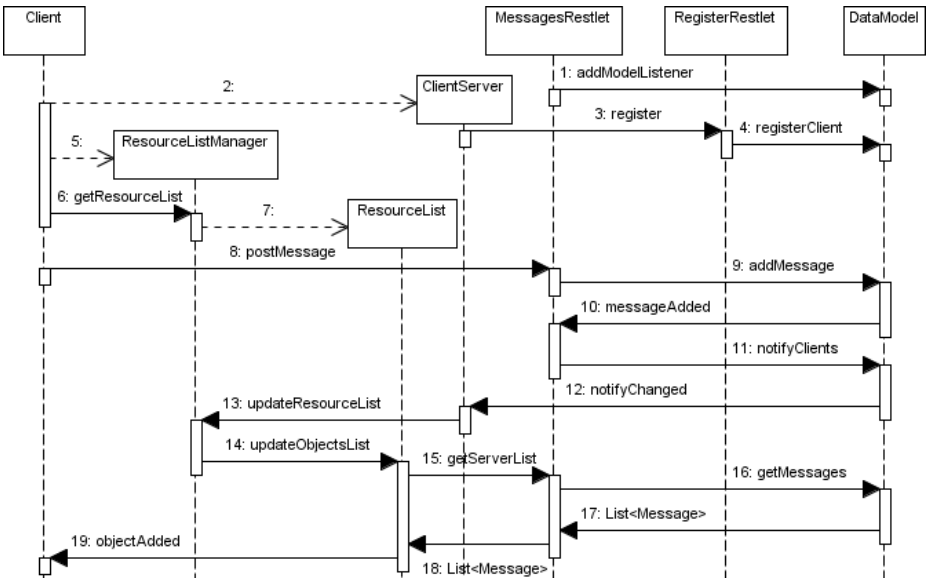

Fig. 8. UML sequence diagram illustrating server-client initialisation and communication

implemented in this prototype, comparison of a server data resource with a local copy requires bandwidth proportional to the size of the data resource since each time a client is notified of changes, it downloads the entire resource. This scalability issue can be mitigated RESTfully by providing query-able data resources on the server that return the sequence of changes applied to the resource from a certain time onwards. Clients can maintain the timestamp of their last synchronisation and access this data resource as required, thus reducing network load as well as the synchronisation effort. Moreover, in the context of the MiGen system, the data resources that are necessary to support the CD/ST and eGeneraliser clients will be relatively small (of the order of 1-100K).

5.2 Realising the MiGen System Architecture

It is now possible to illustrate how the various components presented in Section 4 can communicate using the infrastructure described above. Fig. 9 begins with a student manipulating their constructions and expressions within their eXpresser User Interface, which updates the data structures managed by the corresponding eXpresser Information Layer 1. This posts these events to the MiGen server 2, which then notifies the ClientServers of interested parties (the eGeneraliser server in this example) that the relevant data resources have changed 3. The eGeneraliser server continues to receive notifications of the student's actions until such time as it infers that a landmark has occurred which requires some feedback to be given to the student. At this point, it posts an update to the learner model and student feedback information stored in the MiGen Database 4. An instance of the CD/ST Information Layer, which has previously registered its interest in receiving information relating to the updated learner model attributes of this

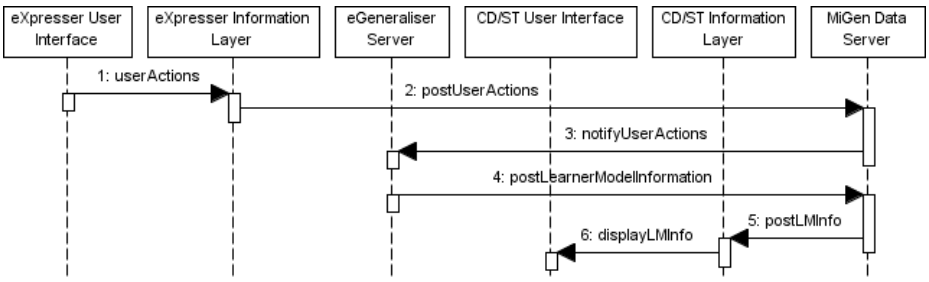

Fig. 9. System sequence diagram

student (by creating a ResourceList pointing at the appropriate URL), is then notified 5, updating its corresponding user interface as appropriate 6.

Fig. 10 shows two functional representations for the CD/ST User Interface which have been implemented as part of our proof-of-concept. Fig. 10a shows a CD representation based on a layout of the classroom in which the position of each student is shown, and each student is accompanied by a 'traffic light' graphic indicating the status of some aspect of their learner model, as selected by the teacher to view. In this example, it may be the TaskShortTermModel attribute 'is the student on-task?', which can take one of three values — yes/no/maybe; these values are visualised as red for 'no', green for 'yes' and amber for 'maybe' (the latter indicating uncertainty on the part of the system).

Fig. 10b provides an informationally-richer ST representation which displays the progress of a set of students with respect to a set of landmarks, as selected by the teacher. Both of these representations update in real-time in response to information posted to the DataModel via the clients. For the purposes of the prototype, the representations are based on messages that clients submit. Despite our prototype not yet encompassing the functionality of the eGeneraliser and MiGen Data Server, the communication mechanism will be identical with these components in place.

Our prototype provides crucial functionality required by the architectural design of the MiGen system: implementing the 'data accessor' API using Restlet and the

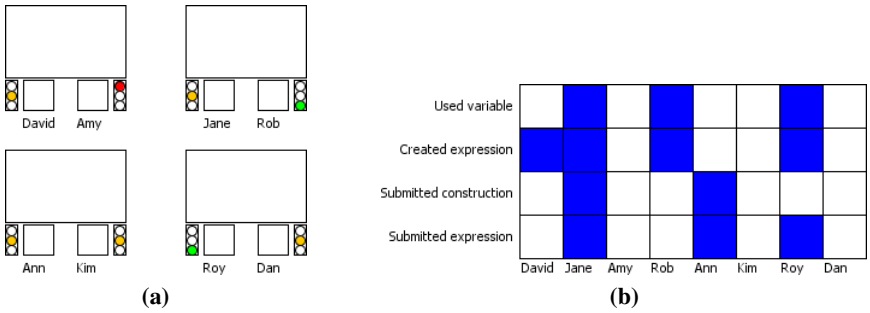

Fig. 10. Prototype CD/ST visualisations. **(a)** Classroom view; **(b)** Matrix view.

ResourceList infrastructure. Following on from the development of this prototype, the next steps are to extend the Restlet-based architecture so as to implement the full server-client architecture of the MiGen system. This will involve determining the overall set of data resources that will be required, designing and implementing the underlying MiGen database and implementing the various tools' servers. This will be followed by integration of the exisiting eXpresser UI and Information Layer with the eXpresser server, and development and integration of the other UIs and Information Layers.

6 Conclusions

We have described the conceptual modelling and architectural design of the MiGen system, which aims to support 11–14-year-old students' learning of mathematical generalisation. MiGen is aiming to provide students with personalised feedback based on their recent actions rather than just on their knowledge levels, aiming to balance students' freedom to explore and discover with sufficient support to foster progressive building of knowledge. Teachers will be assisted in monitoring their students' activities and progress by appropriate visualisation and notification tools, in order to support the teacher in focussing her attention across the class and in formulating her interventions.

The Conceptual Model presented in Section 3 has been derived after a considerable requirements elicitation effort and represents a significant step towards the development of a common vocabulary and understanding for the multiple disciplines engaged in the MiGen project. It also underpins the ongoing logical design of the MiGen database. The System Architecture presented in Section 4 is simple, modular and scalable, aiming to meet the project's objectives of incremental development and evaluation of the various tools of the system, and also easy deployment in schools. In Section 5 we described a working proof-of-concept implementation of the architecture and we discussed how this prototype will be extended in order to implement the first full version of the MiGen system. In the coming months, the project team will continue with iterative design, development and evaluation of the various tools of the system, continuing to follow the methodology outlined in Section 3 and aiming for the first full system evaluation to take place in Autumn 2009 with groups of students and their teachers.

Acknowledgements. The MiGen project is funded by the ESRC/EPSRC Technology-Enhanced Learning programme (RES-139-25-0381). We thank the other members of the MiGen team for our ongoing collaborative research on the project and for many stimulating discussions.

References

1. Küchemann, D., Hoyles, C.: Investigating factors that influence students' mathematical reasoning. In: PME XXV, vol. 3, pp. 257–264 (2001)
2. Healy, L., Hoyles, C.: A study of proof conceptions in algebra. Journal for Research in Mathematics Education 31(4), 396–428 (2000)
3. Mason, J., Graham, A., Johnston-Wilder, S.: Developing Thinking in Algebra. Paul Chapman Publishing, Boca Raton (2005)

4. Geraniou, E., Mavrikis, M., Noss, R., Hoyles, C.: A learning environment to support mathematical generalisation in the classroom. In: Proceedings of CERME 6, Lyon, France (2009)
5. Lesh, R., Kelly, A.E.: A constructivist model for redesigning AI tutors in mathematics. In: Laborde, J. (ed.) Intelligent learning environments: The case of geometry. Springer, New York (1996)
6. Kynigos, C.: Insights into pupils' and teachers' activities in pupil-controlled problem-solving situations. In: Information Technology and Mathematics Problem Solving: Research in Contexts of Practice, pp. 219–238. Springer, Heidelberg (1992)
7. Hoyles, C., Sutherland, R.: Logo Mathematics in the Classroom. Routledge, New York (1989)
8. Kirscher, P., Sweller, J., Clark, R.E.: Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiental and inquiry-based learning. Educational Psychologist 41(2), 75–86 (2006)
9. Mavrikis, M., Geraniou, E., Noss, R., Hoyles, C.: 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 2008 (2008)
10. Cocea, M., Magoulas, G.: Combining intelligent methods for learner modelling in exploratory learning environments. In: Proceedings of the International Workshop on Combinations of Intelligent Methods and Applications at ECAI 2008, pp. 13–18 (2008)
11. Cocea, M., Gutierrez-Santos, S., Magoulas, G.: Challenges for intelligent support in exploratory learning: the case of shapebuilder. In: Proceedings of the International Workshop on Intelligent Support for Exploratory Environments at EC-TEL 2008 (2008)
12. Thompson, P.W.: Mathematical microworlds and intelligent computer-assisted instruction. In: Artificial intelligence and instruction: Applications and methods, Boston, MA, pp. 83–109. Addison-Wesley Longman Publishing Co., Inc. (1987)
13. Hoyles, C.: Microworlds/schoolworlds: The transformation of an innovation. In: Keitel, C., Ruthven, K. (eds.) Learning from computers: mathematics education and technology, pp. 1–17. Springer, Berlin (1993)
14. Gutierrez, S., Mavrikis, M., Pearce, D.: A learning environment for promoting structured algebraic thinking in children. In: Proceedings of ICALT, Santander, Spain (2008)
15. Noss, R., Hoyles, C., Mavrikis, M., Geraniou, E., Gutierrez-Santos, S., Pearce, D.: Broadening the sense of 'dynamic': a microworld to support students' mathematical generalisation. The International Journal on Mathematics Education (ZDM) 41(5) (to appear, 2009)
16. Pearce, D., Poulovassilis, A.: Architecture specification of MiGen v1 (April 2009) Version 0.2., `http://www.migen.org/publications/reports/arch-spec/0-2`
17. Fielding, R.T.: Representational State Transfer (REST). In: Architectural styles and the design of network-based software architectures, ch. 5. University of California, Irvine (2000); PhD Thesis