Feedback Authoring for Exploratory Activities: the case of a Logo-based 3D Microworld

Sokratis Karkalas[1], Manolis Mavrikis*[1], Marios Xenos[2] and Chronis Kynigos[2]

[1]UCL Knowledge Lab, UCL Institute of Education, London WC1N 3QS, UK
[2]Educational Technology Lab, National Kapodistrian University of Athens

Keywords: feedback authoring, exploratory learning environments

Abstract: This paper presents AuthELO an authoring environment that can be used for the configuration of logging and authoring of automated feedback for exploratory learning objects (ELOs). ELOs are web components (widgets) that can be integrated with learning platforms to synthesise highly interactive learning environments. AuthELO has been developed in the context of the MCSquared project that is developing a platform for authoring interactive educational e-books. This platform comprises an extendable set of diverse widgets that can be used to generate instances of exploratory activities that can be employed in various learning scenarios. AuthELO was designed and developed to provide a simple, common and efficient authoring interface that can normalise the diversity of these widgets and give the ability to non-experts to easily develop or customise the feedback that is provided to students using a data-driven approach. In this paper we describe the architecture and design characteristics of AuthELO and a small-scale evaluation with activities in a logo-based 3D microworld called Malt+. We reflect on both the challenges of the authoring process and the pedagogical potential of the feedback when these activities are used by students.

1 INTRODUCTION

Authoring educational interactive tasks is a challenging and time consuming endeavour particularly if they include some form of adaptive or intelligent support to the learner. While there is an abundance of tools that allow non-expert developers, such as educational designers or teachers, to author their preferred activities, these are limited to static content or to pre-defined question-answer activities. As we review in Section 2, researchers in the field of Intelligent Tutoring Systems are looking into the development of tools that ease the authoring process for ITS but have largely remained in the realm of structured interaction. We are interested in highly interactive, exploratory activities that take place within open learning environments including microworlds (Healy and Kynigos, 2010; Mavrikis et al., 2013b). Although such environments can be effective in supporting learners' development of conceptual knowledge, they require significant amount of human or computer support (Mavrikis et al., 2013a). Despite the fact that research in the area has demonstrated that it is possible to delegate part of this support to intelligent components (e.g. Bunt et al. 2001; Gutierrez-Santos et al. 2012), there have been little attempts to reduce the entry threshold for both programmers and end-users (Blessing et al., 2007).

This paper presents AuthELO, a tool for authoring exploratory learning objects (ELOs), configuring the logging and programming the automated feedback they provide. This tool has been developed in the context of the Mathematical Creativity Squared (MCSquared) EU-funded project (http://mc2-project.eu/) that is developing a platform for authoring interactive educational e-books. This platform comprises an extendable set of diverse widgets that can be used to generate instances of exploratory learning activities that can be employed in various learning scenarios. In this project we designed and developed a tool that is able to provide a simple, common and efficient authoring interface that can normalise the heterogeneity of these widgets and reduces the time it takes and the skills required to program the feedback that can be provided to students based on their interaction.

Section 3 presents the development methodology that underpins the design of AuthELO. Sections 4 and 5 present the architecture and the tool in detail. Section 6 presents an evaluation of the current prototype with a logo-based microworld (Malt+) and Section 7 concludes the paper and briefly discusses our next

*Corresponding author m.mavrikis@ucl.ac.uk

steps.

2 RELATED WORK

The development of learning material that is interactive and provides automated intelligent feedback to the students falls naturally into the category of ITS authoring systems. There have been many such systems developed in the past. Database-related tutors like SQL-Tutor, EER-Tutor and Normit (Mitrovic, 2012) are cases that follow the constraint-based modelling approach. To author web- and constraint-based tutors Mitrovic et al. (2009) developed ASPIRE. The use of simulation-based authoring is presented in Munro (2003). An approach that is used for the development of adaptive hypermedia is presented in Brusilovsky (2003). An attempt to lower significantly the skill threshold required is the model-tracing approach (Blessing et al., 2007). Most of these approaches, although different, they converge in that they all presuppose the use of low level technical expertise for the authoring. Systems that require no programming include the ASSISTment Builder (Razzaq et al., 2009) and Redeem (Ainsworth et al., 2003). The latter is an approach that combines existing material with teaching expertise to develop simple intelligent ITSs. A mixed system that supports the development of two types of ITSs is CTAT (Aleven et al., 2009; Koedinger et al., 2004). It supports the development of cognitive tutors and example-tracing tutors. The latter case requires no programming at all.

All of these systems are typically domain-specific solutions that may require low level technical expertise and usually offer fairly limited and not easily generalisable output. That seriously limits the applicability of these tools to a wider range of learning scenarios. One of the most recent developments in the field is the Generalized Intelligent Framework for Tutoring (GIFT) (Sottilare et al., 2012) that provides tools to support various elements of the authoring process. Although GIFT targets domain experts with little or no knowledge of computer programming or instructional design, at the moment it mostly enables rapid development of expert models and other domain knowledge. This results in a fully-fledged ITS that depends on the services provided by GIFT. That may limit the re-usability of the authoring tool with other learning platforms or may be beyond what is needed or what is possible with limited resources (e.g. of a teacher wanting to adapt a simple activity). At a conceptual and architectural level our system resembles SEPIA (Ginon et al., 2014). SEPIA is designed so that automated support can be added in the form of an epiphytic application that is external to the learning environment. Integration does not require changes in the target environment and interoperation is not based on domain specific models and tools.

From an end-user pespective, the most relevant solution to our approach, and the one that seems to require the least amount of cognitive load for the author, is the example-tracing approach (Aleven et al., 2009). The author develops feedback by executing the activity like a student. This provides the author with a tree-like view that is representative of the current state of the student. The author can then annotate the diagram and determine the behaviour of the tutor. The disadvantage of this approach is that it is domain-specific and not generalisable beyond structured tasks that have a relatively limited range of possible alternative paths. In an exploratory learning environment these paths are potentially infinite.

In our system we follow the example-tracing approach but we are not using the visualisation part simply because it is impossible to represent visually all the possible states in such diverse domains and open environments. Our tool must be generic enough so that it can be used with exploratory learning environments. Support is expected to be task-dependent but tasks may not be structured. Authoring must be based on data that becomes available as the student interacts with the environment. The author generates data and utilises this information in order to form sensible rules for the generation of feedback. These rules are currently expressed through programming but our intention is to provide a service that can be accessible through different levels of specificity. That will make the system usable by authors with different levels of expertise without compromising the ability to intervene at the lowest level if necessary. A high level language that is specialised in feedback authoring and a visual programming shell will be the high level constructs that will make it easily accessible to non-technical users.

3 METHODOLOGY

In this project the main objective is to design and develop an authoring tool for the engineering of automated (intelligent) support for online learning activities. As mentioned, we are interested in highly interactive 'widgets' that can either be standalone activities or live in the context of an e-book. Such widgets offer learning opportunities through exploration and discovery of knowledge in an unstructured manner.

The methodology we have followed for the design and development of AuthELO is based on previ-

ous work presented in (Gutierrez-Santos et al., 2012). This approach is based on the premise that the complexity of the task can be reduced and made manageable through the compartmentalisation of different concerns regarding the different aspects of the problem. In practice this can be done by focusing on the three most important questions related to support:

- What is the situation now? (evidence)
- Which aspect needs support? (reasoning)
- How should the support be presented for maximum efficacy? (presentation)

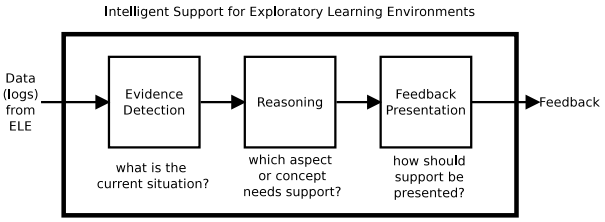

Figure 1: Conceptual data flow of support for exploratory learning. From Gutierrez-Santos et al. (2012)

Each one of these questions corresponds to different aspects of the problem and thus may require different approaches and expertise. Considering these aspects separately reduces the skill threshold required to deal with the problem in its entirety. Typically, in this scheme, the development of support moves towards the opposite direction of the data flow. Designers would start from the presentation and the process would gradually move towards the development of components that produce evidence. In this project the presentation is designed based on the assumption that an exploratory learning system should not intervene in the process in an intrusive manner (Mavrikis et al., 2013a). Support should not be provided in order to manipulate the students and control their behaviour. The system should be discreet and inform the users for potential issues but not interrupt the learning process. On the other hand support should always be available on demand. Students may not be able to exploit the full potential of such learning environments if there is not enough support available to direct them (Mayer, 2004; Klahr and Nigam, 2004). In this tool support is provided after the student initiates the process. We also provide the learning platform the ability to use the same functionality in order to display informative messages to the users regarding the current state of the activity.

The focus of this work is on reasoning and the acquisition of evidence that can support it. For the former we collected a number of use cases of specific learning activities developed in GeoGebra [2], Malt+ [3] and FractionsLab [4]. Expert designers and educators provided us with complete usage scenarios for each activity that include potential student misconceptions, landmarks that can indicate important states of the constructions and the respective feedback that the system is expected to provide to students. This information helped us form the initial requirements for the reasoning part and they have also been transformed into batteries of tests for the technical evaluation of the software.

The data acquisition part comes after because it depends on the reasoning part. Having all the information about the needs of the reasoning part enables us to identify the requirements for the evidence part. The challenges we identified for that part follow:

- need for methods to make the widgets generate the required data
- need for methods to transfer this data between tiers
- need for methods to efficiently store that data in the tool and make it processable so that it can be used for answering queries to the reasoning part

4 ARCHITECTURE

The tool is a native HTML5 application with no external dependencies and is physically decoupled from learning platforms. It does not implement any platform-specific or proprietary functionality and therefore its service is not limited to an existing platform. It has been designed in a way so that its functionality can be provided in a service oriented approach using standardised communication protocols and data formats. After the tool is virtually integrated with a learning platform from the users' perspective the whole system looks unified and homogeneous. Integration is seamless and requires nothing more that setting up a url along with the parameters that provide information on how to instantiate the learning object to be configured and where to store the configuration data. This information is stored in the learning platform and is used whenever an author wants to configure logging and automated feedback for a learning object.

The author initiates this process in the learning platform and implicitly gets redirected to AuthELO. From that point on the tool takes over. It creates an

[2] https://www.geogebra.org/

[3] http://etl.ppp.uoa.gr/malt2/

[4] http://fractionslab.lkl.ac.uk/

instance of the widget that lives in its own private and secure space (sandbox). The two software components operate as independent applications in parallel (asynchronously) within the same browser instance. The glue between them is another component that is called Web Integration & Interoperability Layer (WIIL). WIIL as the name suggests, is a web component that can be used to integrate other web components with a platform. It can also provide a simple yet efficient communication mechanism so that the integrated components can be interoperable. This is described in (Karkalas et al., 2015b). An earlier version of the WIIL and its potential usage was presented in (Karkalas et al., 2015a).

Upon instantiation, the widget sends to AuthELO its widget-specific metadata. That is information about the types of elements that can exist in the widget environment and the types of events that these elements can generate. This data is maintained in local in-memory databases [5] at the AuthELO side of the browser. AuthELO uses this information to construct dynamically a graphical user interface for the configuration of logging. Something that needs to be noted here is the dynamic nature of this process. There are no presumptions about the information that is received from the widgets. Different widgets may provide different metadata and that, in turn, may result in the formation of different interfaces.

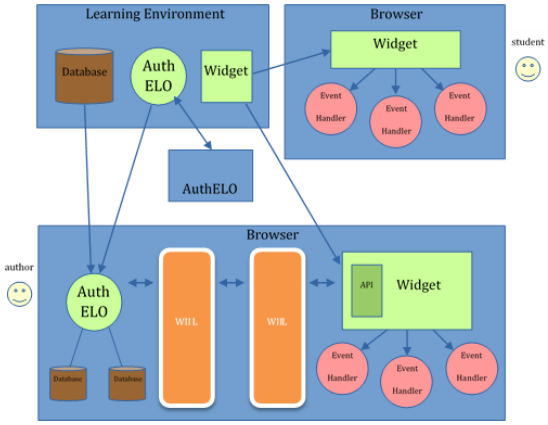

Figure 2: AuthELO's architecture

This GUI is immediately usable by the author. The author can set up data logging rules that have immediate effect on how the instance behaves. The rules are stored in local databases in AuthELO and they are also sent to the widget so that the respective event handlers can be registered. After registration, the widget is able to generate data according to what the author prescribed in the configuration interface. This data becomes directly available to the author for inspection through the WIIL (see activity log in fig. 2). The author uses this information to make decisions about what feedback needs to be provided to the students (fig. 8). This part of the configuration is also stored in local databases.

The logging and feedback configuration is then passed by the tool to the learning platform, so that these settings can become available to the actual widget instances that are going to be used by students. The learning platform sends these settings as part of the initialisation parameters during the widget launch process.

[5]We used TaffyDB (http://www.taffydb.com/) — an open source, lightweight and efficient NoSQL database

5 THE AUTHELO TOOL

AuthELO is designed to provide a generic interface between web-based learning objects, learning platforms and authors that want to synthesise exploratory learning activities with them. The design is based on the following five main requirements:

- Authors must be able to dynamically configure what data will be logged by a learning object during a session with a user. This can be data generated from interactions between the user and the widget and derivative data that gets generated by the widget itself as a result of some event.
- Authors must be able to specify rules about real-time feedback that should be provided to the students. These rules should be based on log data that is dynamically generated as the student engages with the activity.
- It should be possible for authors to configure all the available widgets through a common interface. This interface should be able to hide the diversity of potentially heterogeneous learning components that might be offered in the system.
- The tool must not impose barriers in terms of skills and technological expertise. Teachers with a certain degree of IT literacy should be able to use it for authoring of interactive learning material.
- The tool must be able to offer opportunities for exploratory authoring of feedback reducing the cognitive load that is expected for non-structured tasks of exploratory activities.

The general aim of this project is to provide a tool that offers the following:

- It is simple to use
- It can be used with diverse learning components

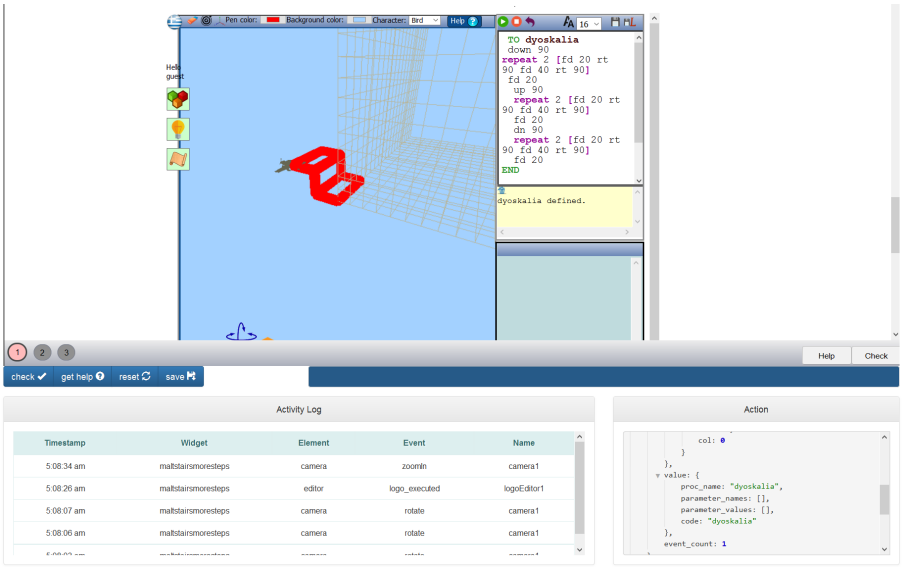

Figure 3: The authoring interface provides a live instance of the activity along with a toolset for testing and debugging.

- It can be used effectively to configure feedback for non-structured tasks in exploratory learning environments

5.1 THE AUTHORING INTERFACE

When the tool gets instantiated it looks like Figure 3. The authoring interface is provided as a triplet of tabs named 'widget', 'logging' and 'feedback'. The first page (widget) provides a visual of the activity along with a basic toolset that can be used for testing and debugging. In the middle of the page there is a live instance of the widget that represents the activity that is going to be presented to the student through the learning platform. The author can interact with it in the same way that a student would and experiment with different configurations until the result satisfies the learning objectives that have been set. During this interaction the author can see what data gets generated and display messages useful for debugging.

5.2 AUTHORING LOGGING

Configuration options for logging are given in the homonymous tab-page. This page is dynamically constructed by the application using information retrieved from the live widget instance that represents the activity.

That means that this part of the tool dynamically changes for different widgets or widgets that contain different constructions. The aim is to for the tool to work with different widget 'instances' i.e. different configurations of the same widget but for different

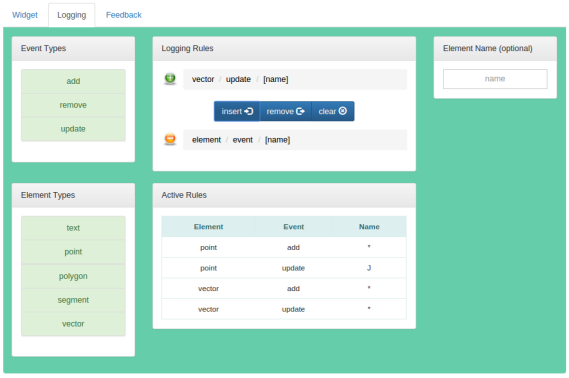

Figure 4: Logging Configuration

activities. The challenge is that these instances contain different types of elements able to generate different types of events. The tool is able to dynamically query the instance and obtain all the information that is necessary to reconstruct itself and adapt to the individual characteristics and needs of the activity. But how can that be possible? Widgets may be third party components that do not provide standard communication interfaces and data formats. So how do we deal with diversity? There is no magic behind this wonderful feature. Communication and interoperability between the tool and widget instances go through WIIL (Karkalas et al., 2015b). In that layer we can reshape APIs and semantically enhance the metadata that is received from instances whenever that is deemed necessary. That takes care of diversity but what happens if the initial construction that is given for the activity does not contain all the necessary elements and

events? The assumption is that the tool queries the live instance and retrieves information about what is currently present in the construction. For that part there is no easy answer. You either get the widget implementers to expose a method that provides information about all the possible elements and events for that particular widget or you get the activity author to create extra elements that may need to be recorded in the log files and hide them from the user. In this particular implementation we followed the second approach simply because it would be impossible to force widget vendors to change their implementations and it would be impractical to visualise endless lists of element and event types that would not be used in the activity. Cluttering the authoring interface with unnecessary information would compromise the usability of the tool.

In this page the author can see a list of the element types that are supported by the widget along with the types of events that these elements can generate. These lists are presented as sequences of buttons. The author can press a button and select an element or an event type. When that happens the name of the selected entity appears in the 'logging rules' section next to the plus sign.

The rules have immediate effect on how the instance behaves. They are stored in local databases in AuthELO and they are also sent to the widget so that the respective event handlers can be registered. WIIL takes care of the underlying operations for that. After registration, the widget is able to generate data according to what the author prescribed. In Figure 5 we can see where new rules are formed. A combination of an element type with an event type gives us a valid rule. The author can optionally provide a name for a specific element if needed. If the rule is ready, it can be inserted by pressing the button 'insert'. The rule in Figure 5 instructs the system to generate events when the value of the point element 'A' changes.

If we want to generate update events for any element of a point type then we can omit the name. If we attempt to insert a more generic rule than one that already exists then the system will give us a warning message but it will allow the operation.

The opposite is not true. If we attempt to insert a more specific rule than one that already exists then the system will not perform the operation because it will not have any effect at all.

If the rule is exactly the same as an already existing one the system will reject it. If a rule needs to be removed then the author can select it by clicking on the list in the 'Active Rules' section. The selected rule will then appear in the 'Logging Rules' section next to the minus sign. The rule can be deleted by pressing the 'remove' button.

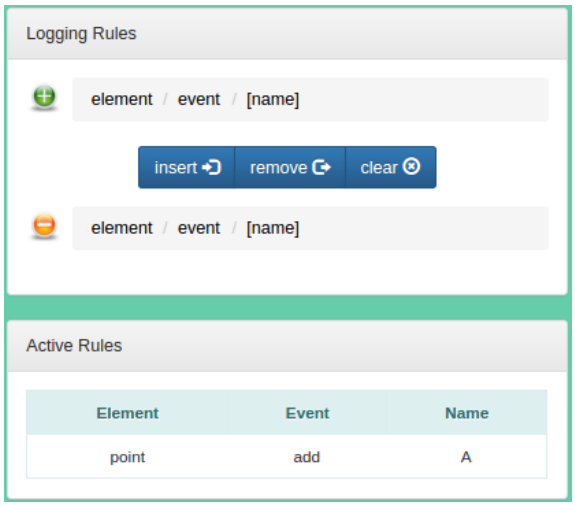

Figure 5: Rule insertion

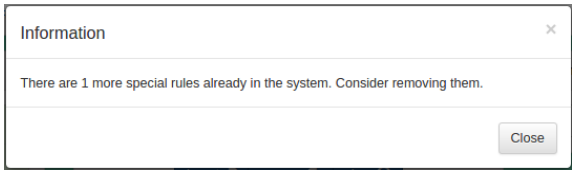

Figure 6: More general rule

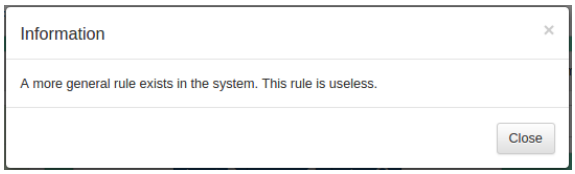

Figure 7: More special rule

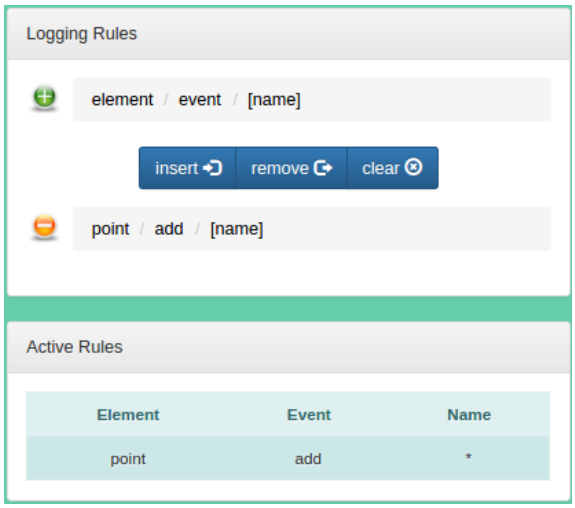

Figure 8: Rule removal

When a rule is inserted it gets immediately acti-

vated. That means that the author can go back to the 'Widget' tab and start generating data by interacting with the widget. The data appears at the bottom of the page.

5.3 AUTHORING FEEDBACK

The configuration or authoring of feedback can be done through the editor that is provided in the 'Feedback' tab-page (see Figure 9). In this part the author can utilise the data generated and displayed in the 'Widget' tab-page and specify rules that state what needs to be done if certain conditions are satisfied. In this version of the tool these rules must be expressed in JavaScript. This is done through a specialised editor that provides support to the author and basic error checking [6]. This way the author can dynamically inject new functionality into the system.

Feedback is presented either through an area under the widget instance or through an intelligent assistant that looks like an owl and displays the message in a bubble (see Figure 10). Authors simply have to change a parameter when they call the function to display the message in order to select one or the other.

After the implementation of feedback rules, the author can go back to the 'Widget' tab and test the feedback. If the author makes a mistake the system displays an error notification under the column named 'System Log' indicating the problem. In this case the changes are not saved and the new functionality is not applied.

Something that needs to be noted here is that this part of the tool is work in progress. We are working towards an intuitive and simple user interface that would not require high-level of programming expertise from an author (particularly a teacher). In the meantime, both in order to test the system but also to ensure that it can be immediately used in the context of the project, we exposed a part of the actual JavaScript code that is used to provide the feedback. This does not affect the usability of the system at this stage, only requires a level of expertise from the author that should be able to at least understand JavaScript syntax.

[6]We incorporated the ace editor (http://ace.c9.io) which is a high performance web-based JavaScript tool. The tool is parameterized to process JavaScript code and display it accordingly. It is equipped with syntax highlighters, automatic code indentation, and code quality control and syntax checking that is based on the well-known tool JSHint (http://jshint.com/)

6 PROTOTYPE EVALUATION

AuthELO is currently being used as the standard tool in the MCSquared platform for the development of automated feedback. MCSquared is an active project and that means that AuthELO is continuously being used and evaluated in practice by a large community of learning designers. The people involved in this process form Communities of Interest (CoIs) and continuously utilise the MCSquared platform to produce new learning material and evaluate it in the classroom. Different CoIs may have a different orientation, specialise in a certain domain and utilise a certain set of ELOs for their productions. In this paper we present the work that has been done by the Greek COI that specialises in constructionist activities with narrative and utilises the ELO Malt+ for the development of learning materials.

6.1 Malt+

Malt+ is an exploratory learning environment that utilises programming to design 3D dynamic graphic models. It consists of a programming editor, a 3D scene and a dynamic variation tool. Learning designers can build domain specific Malt+ widgets (e.g. in mathematics, informatics etc.) that offer students opportunities to explore, redesign and test their own assumptions in order to approach a solution. An avatar on the 3D scene moves and draws shapes and models based on programming code that is executed in the editor. The variation tool can dynamically change the resulting model by changing the programming parameters providing a way to explore modifications in a continual and direct way. As there is a camera available that can change the perspective, activities may exploit the 3D space from the beginning or may start in 2D space and switch to 3D space dynamically. Depending on the activity goals, the presented problem may have more than one solution, it may be solved with multiple alternative programming strategies and as a consequence of those multiple different learning paths may be followed.

A large number of Malt+ widgets have already been produced and studies have examined the functionalities and prospects this environment provides. In (Diamantidis et al., 2015), the study identifies the role of logo programming in meaning generation in mathematical thinking and points out how reconstruction of a program and its 3D result could affect generalization. Another study (Zantzos and Kynigos, 2012) has evidenced that the ability to explore and symbolically represent movements and shapes with Logo, engage students in notions of the conceptual

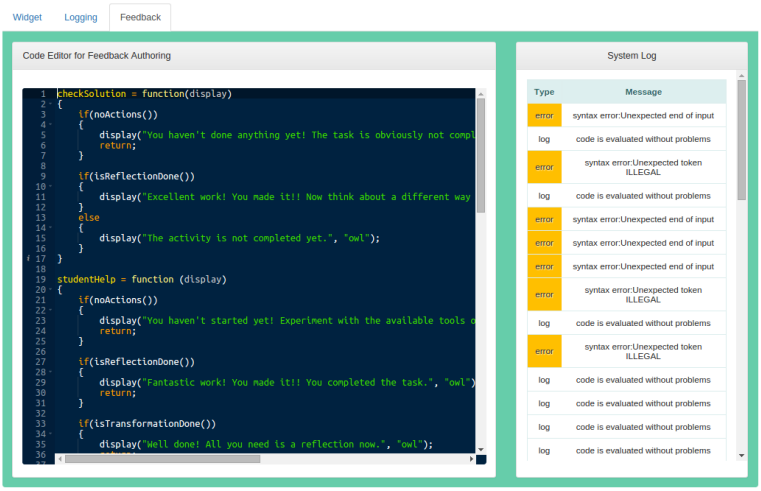

Figure 9: Feedback authoring through a specialised editor and system log

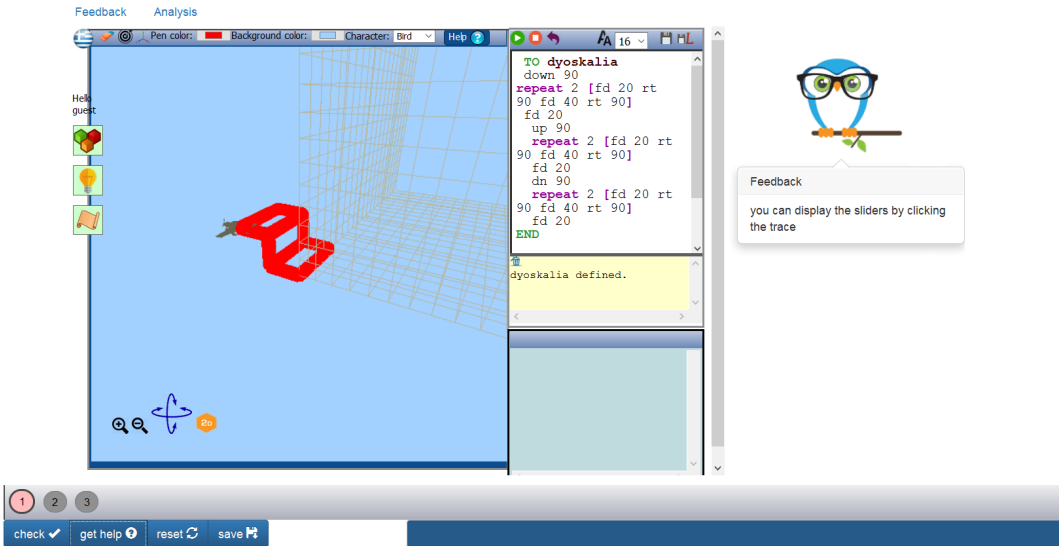

Figure 10: Help messages

field of curvature in space. The role of dynamic manipulation of a 3D shape the environment offers, has also been studied when students exploring the dynamic aspects of an angle (Latsi and Kynigos, 2011).

6.2 The Activities

A typical scenario with Malt+ is to present a 'half-baked' microworld to the students and ask them to amend and/or augment it in order to derive a complete construction or fix problems with the existing one. A 'half-baked' microworld (Kynigos et al., 2007) is incomplete by design, provoking students to build a new artifact that has more meaning for them. Through this process the students engage with the material, the concepts and the techniques involved in an experiential way and discover knowledge themselves. For this evaluation the Greek CoI developed two such activity scenarios named `Staircases` and `Chand Baori`.

6.2.1 Staircases

The e-book `Staircases` puts the student in the position of an architect who designs different types of staircases. This activity is offered in an e-book that provides a series of activities starting from indentifying the characteristics of a stairs and ending with building complex stair types.

This activity comprises two tasks that are interrelated. The beginning of the second part presupposes the completion of the first part. Both tasks have been implemented in the same Malt+ instance. The first part presents a semi-defined (broken) staircase and asks students to explore the staircase parameters, indentify their role in the shape and amend them to build a well-defined staircase. The second part asks the students to find a way to generalise the previous solution so that it can be used for the construction of well-defined staircase with any number of steps. Students use a predefined program to do their experimentations and modify it to provide a solution.

6.2.2 Chand Baori

The activity `Chand Baori` asks students to build a 7-step double staircase as Chand Baori stairs looks like. The activity starts with a small program that draws 2 steps on the 3D space. Students have to indentify the programming code that draws each step, replicate it to draw more steps and find a solution to change direction and move down after the top of the stairs. In this activity depending on the students' programming skills or the target group, totally different strategies can be followed. Thus, a solution may contain only simple repeated commands or loop structures or subroutines or combination of them.

6.3 Challenges

The challenges we were faced with during this process were multifaceted. A technical challenge was to inform AuthELO about the particular set of element types and their respective event types that may be found in a Malt+ environment. Despite the fact that Malt+ is a very sophisticated ELO it does not natively support an API through which this information can be acquired. The solution was to semantically enhance it in AuthELO through WIIL (Karkalas et al., 2015b). For this, a formal description of nine element and four event types was formulated after using Malt+ in various scenarios without any logging rules applied.

Another challenge was to address the requirement of providing feedback for two distinct and interrelated subtasks implemented in the same Malt+ instance. The Staircases activity poses such a requirement. The learner is presented with two questions that have to be followed in succession. The automated feedback component must be able to distinguish between the two and suppress feedback if the first part is not complete. In this case it is expected to direct the learner to finish the first subtask.

6.4 The Evaluation Study

6.4.1 Methodology and Material

In order to design the essential guidelines and find any possible weak points of the feedback system, we conducted two studies in real world conditions: A pilot study and an evaluation study. In both studies we exploited four activities that were designed as exploratory activities with Malt+ widgets. The theme of those activities was 3D staircase design using logo programming code. In the first activity students have to identify the correlation between the different geometrical characteristics of a staircase and modify the logo code to build a generic staircase program. In the second activity they have to modify a logo program to extend a two-step stair to a seven-step staircase. In the third activity the goal is to design a seven-step double staircase (Chand Baori). In the final activity students modify the two-step staircase program in order to draw any double staircase with a given number of stairs. Both studies took place in after school math clubs at a Greek secondary school.

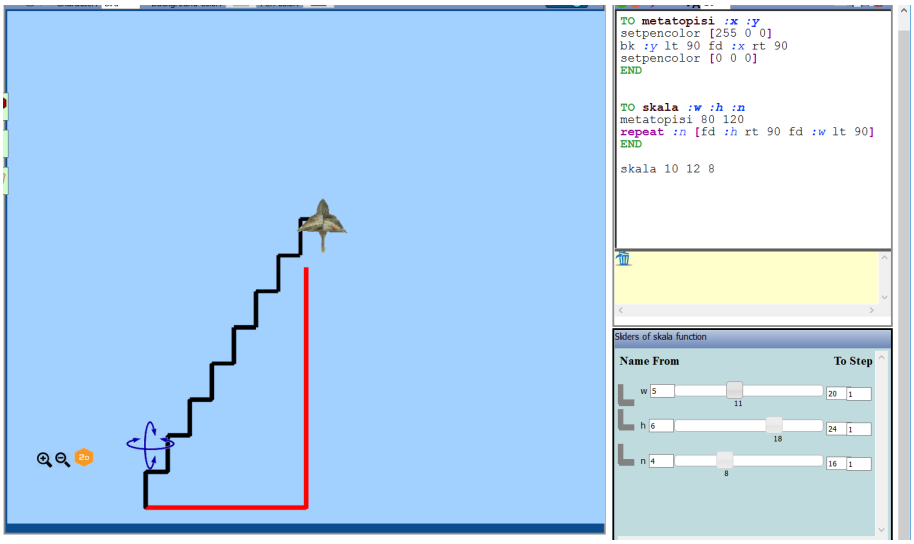

Figure 11: The staircases activity in Malt+

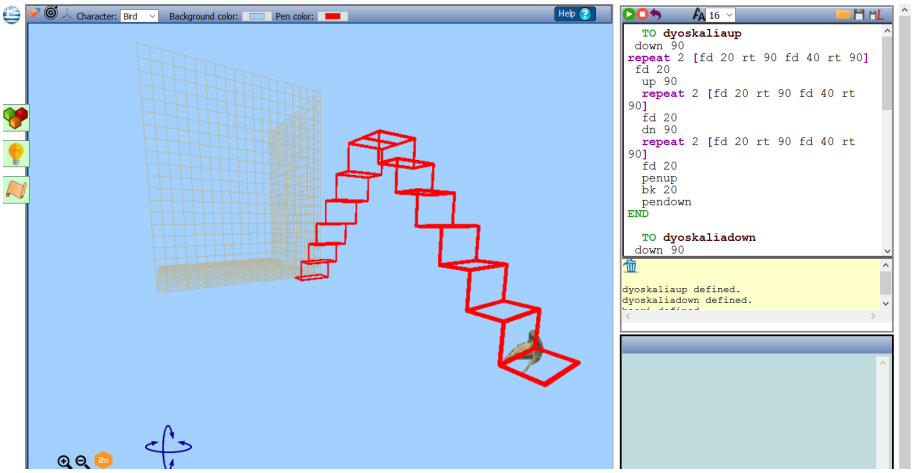

Figure 12: The chand baori activity in Malt+

6.4.2 Design and Pilot Phase

In this phase, based on the main learning designer's experience and feedback from the CoI, we designed an initial set of feedback messages with the goal of getting first an initial understanding of its effectiveness from a pilot study with students. The implementation was first done by a relatively experienced JavaScript developer without previous AuthELO experience. After a preparatory tutorial of twenty minutes he was able to fully develop feedback for the first phase of the evaluation within a working day. Observing the developer using AuthELO we confirmed that, despite the short familiarisation session, he was able to select the items of interest and check directly whether the widget generates the data required. Reflecting on the usage he mostly commented on the ease of authoring thanks to fact that data gets displayed dynamically as he interacted with the widget. As there was no need to consult the widget documentation for anything or to switch context and query the back-end database he confirmed our design rationale that this reduces the overall time, particularly because otherwise one needs to spend a significant amount of time going through the events that generate data — especially in the context of exploratory learning objects. This is not something we can expect the average teacher to have the training or time to do hence the involvement of a developer at this initial phase.

Having designed the feedback, eight students participated in a pilot study, allocated in 4 groups. From this experiment we collected 547 indicators. All students had previous experience with Malt+ environments but they were presented these particular activ-

ities for the first time. The main goals of the pilot study was to:

- identify possible problematic or difficult situations that the intelligent support system should recognise during the activity
- identify sample solution paths the students follow in order to acquire evidence and generalise the designed feedback
- study the students' reactions and the level of acceptance of such a system

After the first iteration, the data collected was analysed along with the tutors' observations and a revision of the feedback was introduced by both the teacher and the original designer. This time it took them approximately two hours to amend the initial design and align it with the new requirements. The effort in this phase focused on providing simple, generic but relevant and helpful feedback. The main advantage of AuthELO as suggested by the teacher taking part in the study was the transparency introduced by being able to see the various feedback rules and feedback messages and discuss them with the original developer. This made him confident that in subsequent iterations he would be able to make modifications through the MCSquared platform himself.

Feedback is always available but it provided on demand. That means that students initiate the process when they feel they need assistance. Due to space limitations we are not describing all the details of the feedback defined but we quote three indicative feedback design decisions for the 'baori' activity:

1. If the time elapsed since the student began interacting with the microworld is less than 30% of the estimated completion time for the activity and the camera has not been used yet the system prompts the student to use it before any other help is given. If the camera has already been used more than 20 times and the sliders are not used then the system prompts the user to try the sliders.
2. If the number of loops identified in the function is more than two then the system gives a hint about possible unecessary code repetition and suggests the user to think about reusability of code.
3. If a function is properly defined but is not called repetitively the system suggests the user to rethink about the reasoning behind the design of the function and how it is supposed to be utilised by the rest of the program. This is obviously related to the previous suggestion (reusability of code).

6.4.3 Testing the Redesign

During the second phase we examined the automated feedback generated both from a technical and from an educational point of view. We also tried to identify the cause of possible failures to provide the required level of help.

From a technical point of view there were no problems experienced during the process. AuthELO was able to express all the different types of feedback required regardless of complexity and the activity player was able to generate the correct feedback at all times.

From an educational point of view the preliminary analysis shows the following:

- Students tend to avoid getting help even if they are encouraged to do so. It is very important therefore to carefully introduce any feedback affordances to students and to discuss with them their purpose.
- Due to the exploratory nature of the activities, students may miss opportunities, get lost and ultimately abandon the activities. Designers, therefore, may need to consider when to design feedback that intervenes in appropriate times as suggested in (Mavrikis et al., 2013a).
- A feedback message that does not meet the 'expectations' of the students could act as an inhibitor of asking further help from the system. It is important, therefore, to design feedback messages carefully and to allow designers and teachers to modify them easily and frequently based on their expertise.

A possible interpretation of the above follows:

- Computer users and especially students are not familiar with the process of asking the system for help. In fact, there is an almost common belief that the system can provide only general help and users have to read a lot of help text to find a useful hint. In our study another factor that may affect the student behavior is that the study sample was part of a special student club where members 'play' with technologies and mathematics and therefore they prefer to explore and try rather than ask the system for ready-made help.
- Students seem to get disappointed very quickly from a given help message that does not seem to be helpful enough. They get a negative 'sense of help' and avoid asking for more.
- It is very hard for the author to identify all possible situations in an exploratory environment so in some cases the feedback may have general guidelines and not specific hints at least initially. Feedback authoring in this case should always be an

iterative process that stops when the level of support in relation to the level of expected support is deemed adequate.

- There are cases, especially during the beginning of the activity, where the author might deliberately not provide very 'targeted' help. Students in this case may perceive this behaviour as a weakness of the system and loose their confidence in it. A possible consequence of that might be that the students ignore further help messages provided by the system because they see it as unworthy of their attention.

We can see how designing automated feedback for an ELO is a difficult and time-consuming process with potentially uncertain results and how an tool like AuthELO facilitates not only the authoring process but also the possibility of easily modifying feedback design and investigating its efficacy.

7 CONCLUSIONS

In this paper we presented a tool that can be used to author automated feedback on ELOs. The tool provides a very simple yet effective interface through which learning designers can configure data logging and authoring rules for feedback. The system is developed as a web-based stand-alone application that is available as a service and can be integrated seamlessly with any learning platform with minimal development or administrative overhead. AuthELO has been thoroughly tested for usability and robustness and its final version is being used as the standard feedback authoring tool for key web-based interactive widgets of the MCSquared project. In this paper we presented an evaluation study that was performed on an e-book that combines multiple instances of the 3D Logo microworld MALT+. The findings of this study suggest that AuthELO satisfies its original design goals as it enables easy development of feedback for complex exploratory learning activities whereas at the same time it is expressive enough to formulate rules for any type of feedback required. According to the study participants, it significantly reduces the cognitive load for both developers and teachers and therefore it seems to lower the entry threshold for potential interested designers who want to create or modify feedback on exploratory activities. The tool can also be used by low skilled teachers with limited or no programming expertise to configure or tweak pre-defined feedback. The findings also suggest that the tool has the potential to enhance author performance and productivity.

In the next version we envisage to provide the same service through a more sophisticated environment that will be a combination of visual programming and a high level language especially designed for expressing feedback. We expect these changes to lower the entry threshold for potential users even further.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement N°610467 - project 'M C Squared'. This publication reflects only the authors' views and the European Union is not liable for any use that may be made of the information contained therein.

REFERENCES

Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B., and Wood, D. (2003). Redeem: Simple intelligent tutoring systems from usable tools. In *Authoring Tools for Advanced Technology Learning Environments*, pages 205–232. Springer.

Aleven, V., Mclaren, B. M., Sewall, J., and Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2):105–154.

Blessing, S., Gilbert, S., Ourada, S., and Ritter, S. (2007). Lowering the bar for creating model-tracing intelligent tutoring systems. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 158:443.

Brusilovsky, P. (2003). Developing adaptive educational hypermedia systems: From design models to authoring tools. In *Authoring tools for advanced technology Learning Environments*, pages 377–409. Springer.

Bunt, A., Conati, C., Huggett, M., and Muldner, K. (2001). On improving the effectiveness of open learning environments through tailored support for exploration. In *10th World Conference of Artificial Intelligence and Education, AIED 2001*.

Diamantidis, D., Economakou, K., Kaitsoti, A., Kynigos, C., and Moustaki, F. (2015). Social creativity

and meaning generation in a constructionist environment. In *CERME 9-Ninth Congress of the European Society for Research in Mathematics Education*, pages 2340–2346.

Ginon, B., Jean-Daubias, S., Lefevre, M., Champin, P.-A., et al. (2014). Adding epiphytic assistance systems in learning applications using the sepia system. In *Open Learning and Teaching in Educational Communities*, pages 138–151. Springer.

Gutierrez-Santos, S., Mavrikis, M., Magoulas, G. D., et al. (2012). A separation of concerns for engineering intelligent support for exploratory learning environments. *Journal of Research and Practice in Information Technology*, 44(3):347.

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

Karkalas, S., Bokhove, C., Charlton, P., and Mavrikis, M. (2015a). Towards configurable learning analytics for constructionist mathematical e-books. *Intelligent Support in Exploratory and Open-ended Learning Environments Learning Analytics for Project Based and Experiential Learning Scenarios*, page 17.

Karkalas, S., Mavrikis, M., and Charlton, P. (2015b). The web integration & interoperability layer (wiil). turning web content into learning content using a lightweight integration and interoperability technique. In *Knowledge Engineering and Ontology Development (KEOD), 7th International Conference on*.

Klahr, D. and Nigam, M. (2004). The equivalence of learning paths in early science instruction effects of direct instruction and discovery learning. *Psychological Science*, 15(10):661–667.

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B., and Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In *Intelligent Tutoring Systems*, pages 162–174. Springer.

Kynigos, C. et al. (2007). Half-baked logo microworlds as boundary objects in integrated design. *Informatics in Education-An International Journal*, (Vol 6_2):335–359.

Latsi, M. and Kynigos, C. (2011). Meanings about dynamic aspects of angle while changing perspectives in a simulated 3d space. In *Proceedings of the 35th Conference of the International Group for the Psychology of Mathematics Education.*, Ankara, Turkey.

Mavrikis, M., Gutierrez-Santos, S., Geraniou, E., and Noss, R. (2013a). Design requirements, student perception indicators and validation metrics for intelligent exploratory learning environments. *Personal and Ubiquitous Computing*, 17(8).

Mavrikis, M., Noss, R., Hoyles, C., and Geraniou, E. (2013b). Sowing the seeds of algebraic generalization: designing epistemic affordances for an intelligent microworld. *Journal of Computer Assisted Learning*, 29(1):68–84.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist*, 59(1):14.

Mitrovic, A. (2012). Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, 22(1-2):39–72.

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., and McGuigan, N. (2009). Aspire: an authoring system and deployment environment for constraint-based tutors.

Munro, A. (2003). Authoring simulation-centered learning environments with rides and vivids. In *Authoring Tools for Advanced Technology Learning Environments*, pages 61–91. Springer.

Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T., and Koedinger, K. R. (2009). The assistment builder: Supporting the life cycle of tutoring system content creation. *Learning Technologies, IEEE Transactions on*, 2(2):157–166.

Sottilare, R. A., Goldberg, B. S., Brawner, K. W., and Holden, H. K. (2012). A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (cbts). In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference*.

Zantzos, I. and Kynigos, C. (2012). Differential approximation of a cylindrical helix by secondary school students. In *Proceedings of the Constructionism 2012 Conference - Theory, Practice and Impact.*, Athens, Greece: National and Kapodistrian University of Athens.