Deliverable N°: D29

Open Student Model

The LeActiveMath Consortium
December 2005

Version 1

Main Authors:
Paul Brna, Nicolas Van Labeke, Rafael Morales, Iain Gibson

Project funded by the European Community under the Sixth Framework Programme for Research and Technological Development

Project ref.no.	IST-507826
Project title	LeActiveMath- Language-Enhanced, User Adaptive, Interactive eLearning for Mathematics

Deliverable status	Restricted
Contractual date of delivery	December 31st 2005 (Month 24)
Actual date of delivery	
Deliverable title	Open Student Model
Type	Prototype
Status & version	1.4
Number of pages	65
WP contributing to the deliverable	WP4
WP/Task responsible	WP4 / T4.3, T4.5
Author(s)	Paul Brna, Nicolas Van Labeke, Rafael Morales, Iain Gibson
EC Project Officer	Colin Stewart
Keywords	

Contents

List of Figures

List of Tables

Listings

Executive Summary

This document provides a description of the implementation of the Open Learner Model (OLM) as required by the workplan for LeActiveMath.

The OLM is one of the component of the Extended Learner Model (xLM), the prototype learner modelling subsystem of LeActiveMath. Its aim it to provide LeActiveMath with an interface for the learners to access, explore and challenge the judgements that the xLM is holding about them. It supports several external representations for displaying the various sources of information that the xLM is using to establish its judgement (belief, evidence, etc.) and a mechanism - loosely based on Toulmin Argumentation Pattern - to control the exploration and negotiation of the beliefs. The OLM is implemented as a SWING applet, using the powerful resources of this Java GUI library in order to provide a suitable and usable interface for the learners.

The implementation of the OLM conforms to the requirements generated by the project, as in Deliverable D5 [11], and to the specification of the Extended Learner Model, as in Deliverable D10 [17]. Justifications are provided for any divergence with both documents. For the reader's convenience, relevant extracts from the Description of Work and from the requirements are summarised in section 1.

The document is organised in four parts:

- Section 2 describes the general principles of the OLM, i.e. which pieces of information – extracted from the Learner Model– are presented to the learner and how they are organised.
- Section 3 describes the architecture of the OLM and its communication with LeActiveMath. For the reader's convenience, information about the integration of the OLM with the front-end of LeActiveMath is also provided in this document. Such information will be replicated, together with similar description of the Learner Model, Situational Model and Learner History components, in Deliverable D31 (*Integration of the Student Model in* LeActiveMath) for an overview of the Extended Learner Model.
- Section 4 describes the Graphical User Interface (GUI) of the OLM, focusing on the external representations used to present the LM to learners, the mechanism used to control the interaction with the model and the mechanism used to support natural language.
- Section 5 concludes the document by highlighting some of the important issues raised by the implementation and deployment of the OLM.

The document contains also a transcription of a user-testing session (section A) and a short description of a tool that has been developed for debugging and testing the Extended Learner Model (the EventGenerator, section B).

1 Introduction

One of the most critical aspects of any adaptative system is certainly how to maintain an accurate model of the users' abilities, representing their preferences, knowledge, aims, needs, skills, misconceptions, etc. Traditionally, learning systems like ITSs have kept these models hidden from the users, implementing them along a "black box" metaphor. But recent developments in the field have seen several systems opening their models the learner, using a "glass box" approach or even literally breaking the glass and actively involving the learner in tuning the model.

There are mainly two reasons for such a move, often refereed as *Open Learner Model* (OLM). The first one is that an OLM is itself a source of learning for the students, by encouraging them to *reflect* on their own knowledge. Some studies have indeed shown that able learners could gain from assessing their own knowledge [3, 1, 18]. Secondly an OLM offers an alternative for *inferential diagnosis* by encouraging the *active* and *explicit* involvement of students (or even peers and teachers) in the diagnosis process and influence the system's judgments. This external involvement is one of the possibilities for "bypassing the intractable problem of learner modelling" [20].

However, many issues still remain to be investigated; for example, how to represent the evolving set of the learner's beliefs about their own state of knowledge [21, 22]; how to give psychological credibility to learner models [2]; how to design systems capable of maintaining learner models in collaborative interactions with learners [5, 6], and how to convey a convincing image of the system as a collaborator [7, 4, 8].

The OLM described in this document is an attempt to answer some of these questions.

1.1 Relevant Parts from the Description of Work

Description This workpackage will include empirical research on how to diagnose and how to react to certain states and reactions of the learner as well as more technological tasks such as the development of an open user model and its components, of the diagnostic components and the integration into LeActiveMath. All components will be software- and user-tested.

Tasks

T4.3 Design, development, and user-testing of the open student model The open student model needs to provide ways of viewing the model, as well as mechanisms that allow the student to be capable of changing the model's contents subject to a negotiation process. The design and development of at least one visualization of the open student model; the definition of those parts of the model that will be viewable; the design and development of the user interface and its navigational and communicational features permitting mixed initiative interaction, drawing on the available services provided by the tutorial dialogue subsystem; the design and development of the negotiation process; the definition of any consequences that should be communicated to the dialogue tutorial management system. Modifications of the prototype following user-tests will be implemented.
Task coordinator: UNN/Glasgow

T4.5 Integration into LeActiveMath The communication protocol and the interfaces to other components of LeActiveMath will be designed, implemented and tested.
Task coordinator: Glasgow

Deliverables

The development process will be incremental, taking advantage wherever possible of small scale user trials to examine significant design decisions as soon as possible. Further evaluation of the open student modelling subsystem will take place before and after evaluation to tune the adaptivity available. Software testing will take place throughout the project's lifetime.

D4.1 Specification of the student model subsystem, interaction history subsystem and open student model subsystem, UNN (Mo 12)

D4.2 Report and prototype of open student model subsystem, Glasgow (Mo 24)

D4.3 Report and prototype of diagnostic functionalities, Glasgow (Mo 24)

D4.4 Integration into LeActiveMath, Glasgow (Mo 24)

1.2 Relevant Parts from the Requirements Analysis

The LeActiveMath requirements analysis [11] contains the following claims about the OLM.

Requirement 5.5	The OLM will be able to present the learner with beliefs about the learner's knowledge, skills, competencies, competency levels, academic interests, media competencies, affective and motivational states.
Supports	The learner thinking about what they know, their interests, preferences, competencies, affective and motivational states, promoting in this way the development of metacognitive skills.
Because	A learner can reflect on the domain being learnt, as well as on their affective state and motivation for learning, acquiring in this way metacognitive knowledge and skills. In addition, the open learner model may be a way for learners helping the system to improve its learner models.
Check-rule	The existence of the corresponding interfaces, as well as qualitative semi-structured interviews which may show the effects on learners.
Issues	The possibility that the learner would consider the learner model too inaccurate, it has to be decided what should be accessible by learners and the extent and mechanisms by they can alter the learner model.

Requirement 5.5 has been met by the definition and implementation of several external representations used to present the content of the Learner Model to the learners (see section 4.2).

Requirement 5.6	The OLM provides the learner with a user interface(s) for displaying and manipulating each of the various aspects of the learner's model properly (e.g. student's domain understanding, affective state, motivation, etc.).
Supports	Focusing the student on different aspects of her learning.
Because	It would be confusing to mix these aspects together and each of these aspects may require a particular mode of representation.
Check-rule	It would be possible to do an experimental study based on two different interfaces
Issues	There are a number of issues in defining how to present the different parts of the OLM and what is an adequate representation.

Requirement 5.6 has been met by defining a mechanism for controlling the exploration and challenge of elements of the Learner Model and implementing an interface specifically supporting the learner in challenging the judgements made by the OLM (see sections 2.2.2 and 4.2.3.1).

Requirement 5.7	The OLM user interface provides mechanisms supporting negotiation with the learner about the beliefs stored in the LM.
Supports	Accountability of the learner model on the face of the learner, as well as its accuracy, helped by the learner. It also supports metacognition.
Because	The system will need to provide evidence justifying the beliefs in the learner model, and vice versa, the learner will need to justify their claims by providing their own evidence. In addition, negotiating the system beliefs, as stored in the learner model, may encourage learners to think more deeply about the nature of domain knowledge and their comprehension of it.
Check-rule	The existence of the facilities for negotiation should be complemented with empirical studies of their effect on the learner model accuracy and trustworthiness, as well as the level of learner engagement achieved by the negotiation and its effects.
Issues	Negotiation is a complex activity, which demands some sort of natural dialogue structure, even if not carried out in natural language. There is also the issue of who is going to have the last word, and the consequences of it.

Requirement 5.7 has been met by the provision of dialogue moves defining each possible interaction acts that the learner can performs and the possible responses from the OLM (see section 4.3).

Requirement 5.8	OLM user interface may hide some of the beliefs found in the LM.
Supports	Preserving the "face" of learners and preventing overload.
Because	Too much "honesty" might be depressing/demotivating.
Check-rule	Experimental study based on two different interfaces.
Issues	Research problem: this is a serious and difficult issue - lots of problems in deciding, in a principled way, what to hide. Also the hiding must be adaptive, but how?

Requirement 5.8 is not fully met in the current implementation of the OLM. As mentioned in the issues of the requirements, this is a research problem on its own; a decision was made to show as much as possible to the learner so that problems and side-effects could be detected and analysed in order to propose hiding solutions. This issue is discussed in section 5.

Requirement 6.6	Communication about LM through natural-language enhanced dialogue.
Supports	Student's inspection and modification of the learner model.
Because	NL-enhanced inspection of LM might be more effective than without NL-enhanced dialogue.
Check-rule	Test how NL dialogue can be more effective than LM inspection without dialogue.
Issues	• Pointing may be more effective than natural language. • From the NLU point of view, lots of complexities involved in using NL dialogue to help LeActiveMath to diagnose the student's state of knowledge or expertise. May require the capability to reason about how the OLM came up with its values in the first place. • Need corpus study to inform the design of such as learner model NL-enhanced dialogue component, and to determine the form of dialogue that we can realistically support. • The diverse content of the learner model will need a wider range of dialogue strategies than normally considered within ITS. • Strategy selection given dialogue context and learner model.

Requirement 6.6 is not fully met in the current implementation of the OLM. Because of cost-effectiveness and easy deployment, a direct manipulation interface (i.e. by pointing) has been preferred to free input (i.e. natural language understanding). Nevertheless, the OLM does support a simple but elegant natural language generation for the output of the interaction. This issue is discussed in section 4.4.

2 Principles of the OLM

Two aspects of the OLM are of importance and described in this section: which information – and under which format – is available from the LM and how this information is to be organised and presented to the learner.

2.1 Extracting information from the LM

The principles and processes ruling the LM have been thoroughly described in both Deliverable D10 and D30 [17, 14] but let's reiterate some of its features, from the OLM point of view.

2.1.1 Beliefs, Belief Descriptors and Levels

The basic idea of the LM is its capacity to build, update and return a portrait of the learner on a couple of inter-related traits. These abilities – referred thereafter as layers of the model – are represented in figure 2.1. They are metacogition, motivation and affect, competency, conceptual and procedural errors (CAPEs) and the domain on which all of the previous layers rely.

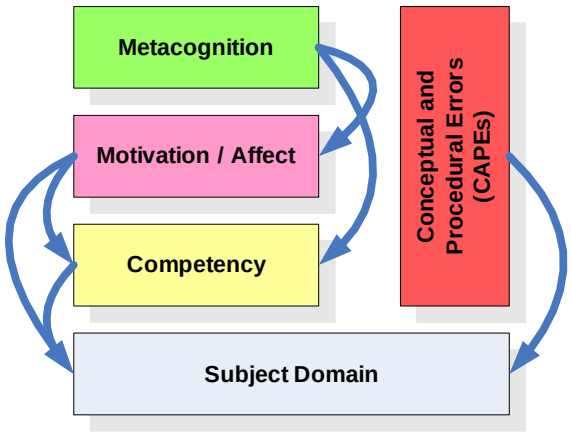


Figure 2.1: The 5 layers of the Learner Model

The portrait of the learner's ability – called belief – is basically an interpretation (or a judgement, as we will refer toi it in the OLM) of the behaviour of the learner according to a given layer. This belief is uniquely identified by its descriptor, internally represented by the n-uplet

```
[domain, capes, competency, motivation, affect, metacognition]
```

complying with the rules of composition described in D10. In the OLM, and when applicable, a NL mapping will be provided to alleviate the obscureness of this internal notation (see section 4.4). Nevertheless, in the rest of the document we will use this n-uplet notation for reference to the belief.

The Learner Model represents beliefs by building a probabilistic distribution on different levels of abilities of the learner. Levels initially comes from PISA and the competencies used in the model

but for consistency, applied to all layers in similar fashion, only their semantics being adapted (the description and semantics of the levels for every layer is given in D30). There are four levels used in the OLM, mapping the variation between low and high ability: `Level I`, `Level II`, `Level III`and `Level IV`. These terms can be seen as a generic placeholder for the Learner Model diagnosis which, in the OLM, will be replaced with a proper natural language terminology, using the phrasebooks associated with each layer of the Learner Model.

Finally, the Learner Model is not merely making suppositions about the likelihood of the learner to be at a given level of ability but is also considering range of abilities, i.e. the learner to be somewhere between `Level I`and `Level III`rather than just at `Level III`. These ranges – or level sets – represented all the possibilities to re-organise the levels, keeping their implicit order. They are represented by their width, i.e. the number of individual levels they include: the singletons (i.e. the sets only containing one of the level), the doubletons (i.e. sets containing two levels such as `Level I-II`), and tripletons (the sets containing three levels such as `Level I-III`). On top of them, let's also mention two special sets: the empty set (i.e. containing no levels at all, also called conflict) and the full set (i.e. `Level I-IV`, also called the total ignorance).

All these terms are important because, under one form or another, they will be displayed in the OLM, presented to the learners and talked about with them.

2.1.2 Beliefs and evidence

Figure 2.2 presents a summary of the chain of subjective decisions/judgments made by the system (LeActiveMath, LM, OLM) from the basic interaction to its final interpretation and presentation to the learner at the GUI level. At the various stages, different procedures are operated, each of them dealing with a particular type of information.

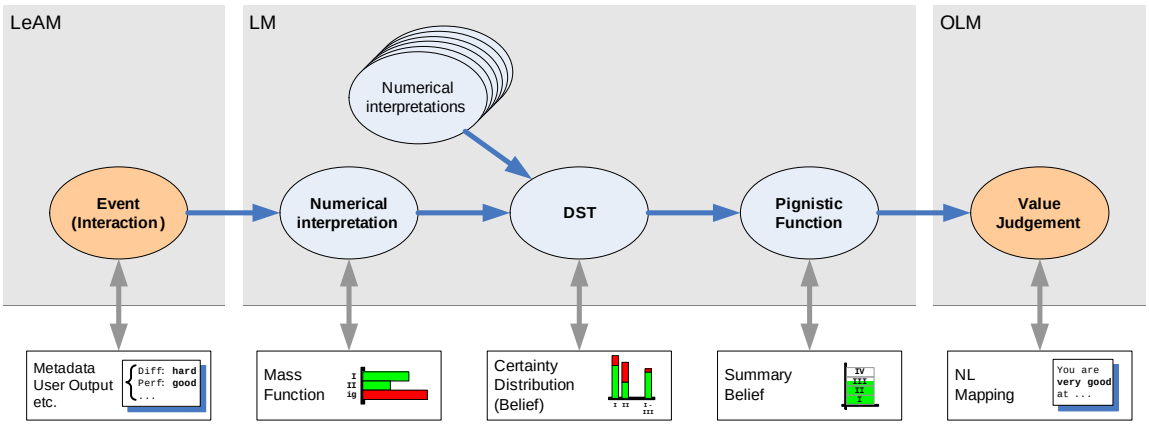


Figure 2.2: Overview of the belief building/updating process of the xLM

Many different types of interaction are currently taken into account by the Learner Model: an exercise is finished by the learner, one particular step of an exercise is performed by the learner, a self-assessment is made by the learner, etc. Each of these events is qualitatively and quantitatively described by various information such as the exercise difficulty, associated competency and competency level, the overall performance of the learner on the exercise, the misconception - if any - associated with an incorrect answer from the learner, etc. [1]

Once this qualitative/quantitative description of the event is sent to the Learner Model, a numerical interpretation of the event is built, internally represented by a mass function, i.e. a distribution

[1] For a complete description of these events and their relevant qualitative and quantitative parameters, refers to the description of the Learner Model, Deliverable D30 [14].

of the assumed learner's ability that could explained the reported performance. This numerical interpretation is now considered by the Learner Model as the evidence for the underlying learner's behaviour.

This evidence is now combined, with any other evidence acquired previously by the system, in order to generate a belief reflecting the overall learner abilities. Internally, this belief is represented by a mass distribution similar to the evidence's numerical interpretation. But to keep the distinction between the evidence (i.e. fact) and the belief (i.e. prediction), we will refer to it as a certainty distribution, i.e. a reorganisation of the mass distribution that emphasises the certainty and plausibility of the Learner Model judgement.

From this broad certainty distribution, the Learner Model can now apply a much narrower decision-making process (called the pignistic function) and finally emits its judgement regarding the most likely ability of the learner: the summary belief. This summary belief is now available for the OLM to map it in a relevant NL pattern to present the learner with a readable value judgement.

2.2 Organising information in the OLM

At one basic level, it can be argued that the only important information – from the OLM point of view – are the summary belief (i.e. the final statement made by the LM) and the facts (i.e. the interaction between the learner and the system) used to reach that statement. But, as it can be seen from the modelling process depicted in figure 2.2, various decisions and interpretations are made at various stages in the process. Therefore, if the aim of the OLM is to support the learner in understanding why and how the Learner Model reached its conclusion, then it becomes important that most – if not all – of these intermediate steps in the modelling process are presented to the learner.

But it means that we need a mechanism to control the delivery of all this information in a way that maintains its significance. In the OLM, this mechanism is inspired by the Toulmin Argumentation Pattern which, besides its simplicity, does indeed provides us with the possibility for managing both the exploration of the Learner Model and the challenge of its judgements. It also supports quite nicely a dynamic reorganisation of the evidence that helps to establish and clarify the justifications presented to the learner.

2.2.1 Toulmin Argumentation Pattern

The Toulmin Argumentation Pattern, proposed by Stephen Toulmin [23] (see figure 2.3), laid out a theory and model of argumentation that went against the accepted model of Formal Logic at the time. It promoted a more abstract, informal approach in dealing with arguments, much closer to daily life.

In brief, *"Data"* (or *"Grounds"*) are the facts and information that are the reason for the claim in the first place — a reasoned beginning; *"Claim"* is the position on the issue, the purpose behind the argument, the conclusion that the arguer is advocating; *"Warrant"* is the component of the argument that establishes the logical connection between the data and the claim, i.e. the reasoning process used to arrive at the claim; *"Rebuttal"* is any exception to the claim presented by the arguer; *"Backing"* is any material that supports the warrant or the rebuttal in the argument; *"Qualifier"* represents the verbalisation of the relative strength of an argument, its soundness.

Applying the Toulmin Argumentation Pattern in the context of the OLM has been be done by the following mapping between each TAP's elements and the Learner Model internal representations (see figure 2.4):

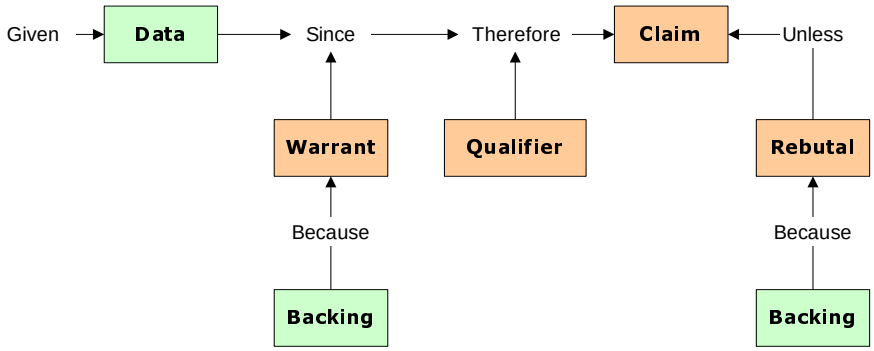


Figure 2.3: Toulmin Argumentation Pattern

- The Claim is associated with the summary belief i.e. a short, straightforward judgement about the learner's ability on a given topic;
- The Data is associated with the belief itself, represented either by its pignistic function (i.e. its simplest internal encoding), its certainty distribution or its mass functions (its most complex internal encoding);
- Warrants are associated with the evidence supporting the belief, represented by their mass distribution. There will be one warrant for every evidence used by the Learner Model to build its current belief;
- Backing are associated with the attributes, both qualitative and quantitative, used by the Learner Model for building the numerical interpretation of the evidence.

In the approach currently implemented in the OLM, we are not considering any Rebuttal, in the sense that all the evidence gathered by the Learner Model is evidence for the learner's abilities. Gathering and manipulating evidence against the learner abilities is an aspect worth investigating but has been ruled out of this project. As for the usage and definition of the Qualifier, it is an issue that we have decided not to consider for the time being – the Learner Model being a probabilistic model, information about certainty and plausibility are explicitly included in the belief (i.e. the data); extracting it in a separate entity (i.e. subject to query and challenge from the learner) seems not to improve the usefulness of the approach.

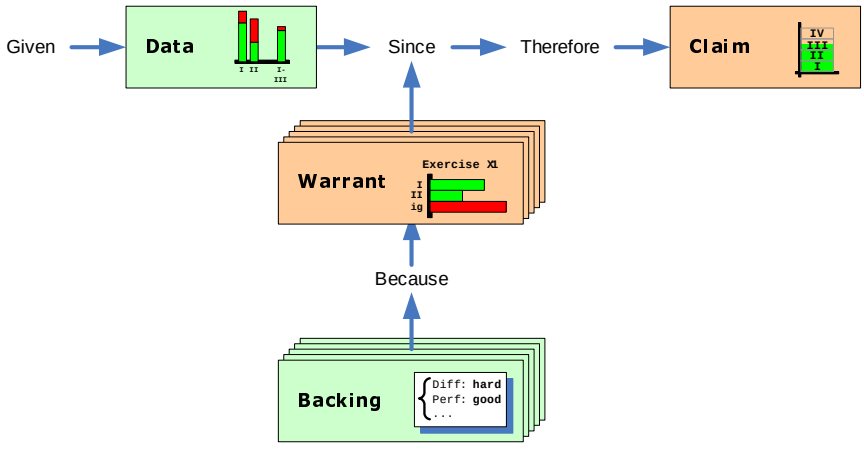


Figure 2.4: Applying Toulmin Argumentation Pattern to the OLM

Consequently, a query to the OLM about some belief can be handed over to the learner according to the following general pattern:

> *Given this certainty distribution and since it is supported by the following list of mass functions, each of them obtained because of what you did/said with item X, therefore I claim that your ability can be described by this summary belief.*

2.2.2 Exploring and Challenging the Learner Model

Toulmin Argument Pattern can be easily and helpfully used to control and contextualise the learner's interaction with the OLM, or to be more precise, with the OLM judgements and justification (i.e. the OLM argumentation). Two aspects of this interaction are of importance here: the exploration of the judgement and its challenge.

A mock interaction between a learner and the OLM can be seen in figure 2.5. It represents various step of exploration of a belief, of its justification to be precise. Every step of the discussion is materialised by the learner expressing his bafflement as why the OLM made its judgement. The initial stage of the exploration takes place when the learner questions the judgement made by the OLM (*"Why do you think I'm* `Level III` *at* `[deriv,think,,]` *?"*). The question is localised clearly on the claim, allowing the OLM to try to justify it by "expanding" the pattern and presenting a deeper justification of the claim, i.e. the data. If the learner is still questioning the judgement, then the data can in turn be expanded to present the first element of the warrant, which can also be expanded to show the reasons (i.e. the backing) for such evidence, etc.

At every stage of the exploration, both the OLM and the learners know exactly what has been questioned, what was the outcome of the question and what remains to be justified.

Toulmin Argumentation Pattern can also help us to understand the source – and therefore the reason – of a disagreement between the OLM and the learner. In the context of such an argumentation pattern, different interpretations of a challenge of the judgement made by the OLM can be considered, depending on the "target"[2] of the challenge.

- When challenging the claim made by the OLM, learners state their disagreement with the overall judgements. Such a challenge is not merely on the evidence used to sustain the judgement but rather on the way they are combined and interpreted, making ground for subjective factors that may not have been – or could not – considered (*"I understand your point, but I still think that you are missing something and that I am not that good on the chain rule"*).
- Challenging a warrant means that the learners disagree with this evidence being used by the OLM to reach its conclusion. The disagreement does not apply to the justification of the evidence itself (as it will be done by challenging the backing) but the presence of the evidence itself in the justification of the belief. This possibility is offered in order to take care of situations that are usually outside of the diagnosis (*"I misunderstood the goal of the exercise, I don't think you should consider it as an indication to my ability"*).
- Challenging the backing of a claim consists in questioning the validity of one – or many – of the attributes used by the OLM to interpret the evidence. It could be a qualitative attribute (*"I don't think that my performance was so low on this exercise"*) or a quantitative (*"I don't think this exercise was so easy"*).

[2] A decent interpretation of challenging the Data has not been devised at the time of writing. Various alternatives have been tried but all of them relied on the explicitation of the processes used to build the belief. This was significantly outside of the scope of the OLM. We therefore decided not to let the Data being subject to challenge, despite the "inconsistency" of the approach.

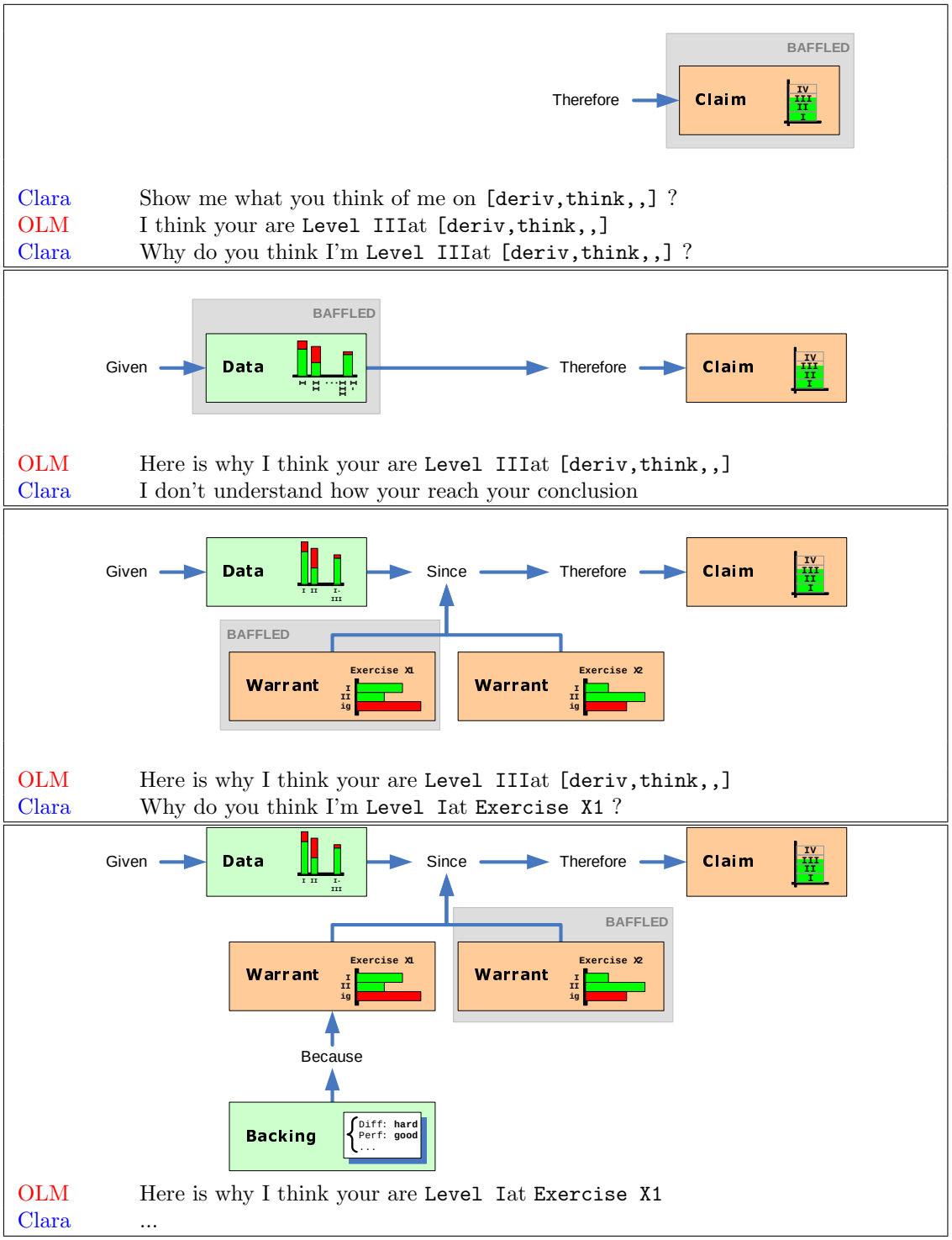


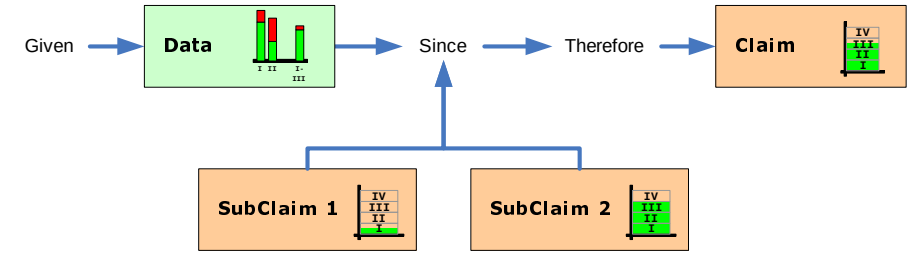
Figure 2.5: Using Toulmin Argumentation Pattern to explore and justifying the OLM

2.2.3 Re-clustering the set of evidence

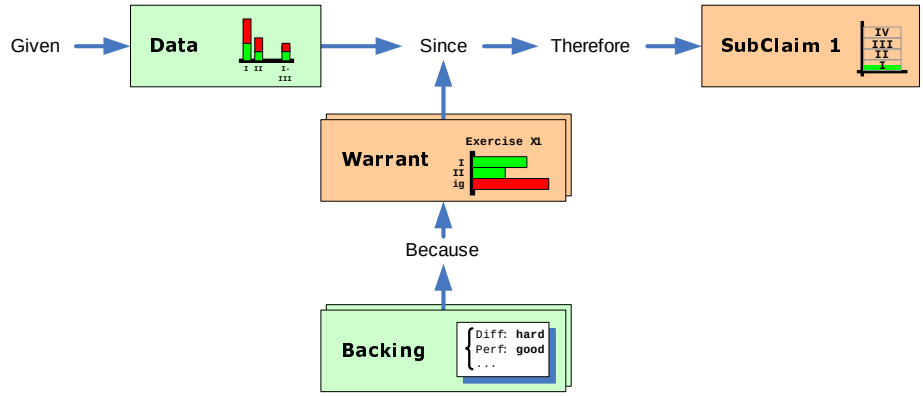
A final interesting property of the Toulmin Argumentation Pattern to exploit is its ability for nesting arguments.

A belief, at its basic understanding, is nothing more than the combination of all the evidence supporting it. The justification of a belief could therefore be seen as just the exploration and appropriation of every piece of evidence, one by one and in the order in which they occurred. But this approach suffers from a very obvious limitation: the sheer volume of evidence that could be potentially gathered by the Learner Model. Hoping that the learner will be able to access and assimilate them one by one is an assumption that will clearly not stand.

One way of solving this problem is to cluster evidence that shares a similar property, altogether into a single judgement, which in turn, can be represented by a Toulmin Argumentation Pattern on its own[3]. The original claim, instead of having several warrants, can now by redefined with sub-claims replacing the evidence being taken care of in this new pattern (as illustrated in figure 2.6).



(a) The Warrant of the TAP is decomposed in sub-claims



(b) Each sub-claim is a complete TAP, supporting the sub-claim by the sub-set of evidence

Figure 2.6: Re-clustering the evidence in a Toulmin Argumentation Pattern

In the current implementation of the OLM, two situations are considered to perform this clustering: the "trajectory" of the learner's ability and the propagation of evidence across the topic maps.

Trajectory Evidence will be collected – and stored – one by one, following the interaction with LeActiveMath they are monitoring. Without pushing our luck too far, it could be assumed that a series of contiguous evidence will all support the same behaviour of the learner, i.e. that they will all indicate a similar trajectory in the evolution of the learner's ability. Therefore, it

[3]Another way, not necessarily contradictory, is to consider the decaying of evidence (i.e. old evidence does not have the same strength as the new ones). Over the time, old evidence will therefore have a smaller and smaller impact on the overall belief and could be just ignored when justifying the judgement.

is possible to decompose the evidence into a set of trajectories and build the justification of the belief on the basis of these changes in trajectory.

Detecting such a shift in behaviour can be easily done by looking at the empty set of the mass distribution of a belief. Also called conflict, information will be distributed in this set as soon as a new piece of evidence does not match with the current belief. In other word, given a particular threshold to compare the conflict with, every piece of evidence gathered up to the evidence introducing a conflict can be organised in a separate claim, representing a given state of the ability (*"I think you are good at the chain rule because you had a first batch of exercises with a quite bad performance but that you improved significantly in the following batch"*).

Propagation Part of the evidence for a given belief will be obtained by the Learner Model because of the propagation of evidence coming from a different topic, connected through one of the layers (e.g. evidence supporting the chain rule will also provide indirect evidence for the derivative). Such evidence (called indirect evidence) will occur at various points in time, interspersed with direct evidence. They can be organised altogether in a proper cluster since they all share the same feature, i.e. supporting a different but connected belief. In this context, a belief about topic X can be justified by the evidence supporting it (evidence that could be clustered according to the trajectories of the abilities) and by the evidence supporting the other belief (*"I think you are good at differentiation because you did quite well on those exercise and because you are good at the chain rule"*).

3 Architecture

Figure 3.1 gives an overview of the architecture of the eXtended Learner Model (xLM) and its connections with LeActiveMath. This architecture has been described in an early deliverable (D10) and this figure has been slightly updated to reflect any changes that did occur during the implementation.

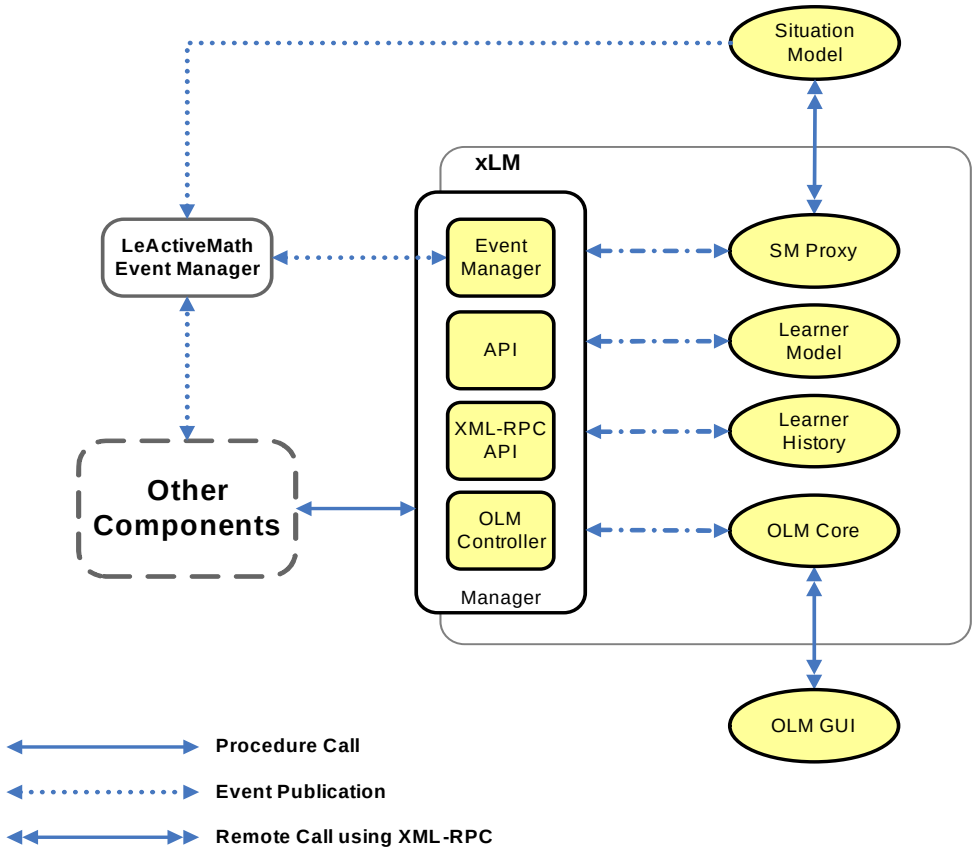


Figure 3.1: An overview of the architecture of the xLM

In this document, we will focus on the OLM part of the xLM.

3.1 Introduction

The OLM is composed of three distinct parts:

- the OLM-GUI, located on the client-side of LeActiveMath and responsible for presenting the collected information to learners and managing their exploration of their model;
- the OLM-Core, located on the server-side of LeActiveMath and responsible for collecting information from the LM and communicating with LeActiveMath;

- the OLM-Controller, also located on the server-side of LeActiveMath and responsible for coordinating the actions of the learner in the OLM-GUI with responses in the OLM-Core.

This decomposition is not surprising, as the implementation of the OLM was made following the architecture recommendations stated in Deliverable D8 [10] and, in particular, the use of a Model-View-Controller (MVC) framework for the front-end components of LeActiveMath.

3.2 The OLM-Controller

The OLM-Controller is a Maverick [1] object whose task is to interpret and handle the relevant HTTP requests, build the appropriate model and pass on the URL parameters – if any.

The Maverick configuration for the OLM (see listing 3.1) describes how the various parts of the OLM are put together.

Listing 3.1: Extract of the Maverick configuration file

```
<command name="olm/show">
    <controller class="org.activemath.xlm.openmodel.OLMAppletController"/>
    <view name="default" path="/main/olm-browser-window.vm"/>
</command>
```

It specifies that the URL associated with the OLM will be of the form

http://localhost:8080/ActiveMath2/olm/show.cmd

that the controller specifically designed for the OLM is called `OLMAppletController` (see listing 3.2) and that the Velocity [2] template to use for generating the view is located in the file `olm-browser-window.vm` (see listing 3.3).

The OLM-Controller itself is a very straightforward object. Upon reception of an appropriate URL, it creates an instance of the OLM-Core (i.e. the model component of the MVC framework) and associates it with a unique URL that will be used to channel the communication between the model and its view (in our case between the OLM-Core and the OLM-GUI). This mechanism is a very important feature of the MVC framework used in LeActiveMath, since it guarantees that each user logged in the system will have their own OLM on the client-side[3].

The OLM-Controller also checks for any parameters added to the URL. In the current implementation, the OLM does support the specification of a belief to initiate the OLM with. This belief is given as a sequence of attribute/value in the URL, where the attribute is one of the identifiers of the appropriate layer of the Learner Model (see section 2.1.1). Thus, an URL like this one

http://localhost:
8080/ActiveMath2/olm/show.cmd?domainId=chain_rule&competencyId=think

will not only open the OLM but will also initiate the dialogue with the learner with the belief `[chain_rule,,think,,,]` (see the description of the STARTUP dialogue move in sections 4.3 for more information about the initialisation of the OLM).

[1] Maverick, see http://mav.sourceforge.net/
[2] Velocity, see http://jakarta.apache.org/velocity/
[3] Let's not forget indeed that LeActiveMath is a web-based and therefore multi-user environment.

Listing 3.2: Extract of the OLM Controller class

```
public class OLMAppletController extends ControllerBase {
    /**
     * Bean properties / Request parameters.
     * Each set function corresponds to a request parameter of the same name.
     * When action() is called, properties are already filled in.
     *
     * The following parameters are used to specify an initial Belief Descriptor to
     * initialise the OLM with.
     * The descriptor is defined by filling in the relevant layer(s) with a valid identifier,
     * ie domainId=derivative&competencyId=think, etc.
     */
    private String domainId = null,
                   metacogId = null,
                   motivationId = null,
                   affectId = null,
                   competencyId = null,
                   capeId = null;

    public String action()
    {
        Map m = super.getModel();
        if(super.getCurrentUser()==null) return sendLoginRequired();

        String learnerId = getCurrentUser().getId();
        String cmdName = getCmdName();

        // check if user is already logged in
        // should be not NULL since the user is obviously logged in
        AppSession appSession = getAppSession(learnerId);

        // But just in case ...
        if (appSession == null)
        {
            // Create AppSession with this user
            appSession = new AppSession(getCurrentUser() , getRequest());
            // Bind AppSession to HttpSession
            bindAppSession(learnerId, appSession);
        }

        // Create the OLM Core from the AppSession and get its URL
        URL url = XmlrpcManager.getInstance()
                        .addServerWithPrefix("olm",new OLMCore(appSession));

        // Put the URL into the relevant parameter of the model
        m.put("olmCoreServiceUrl",url);

        // If available, put the belief descriptor in the parameter
        if (this.domainId!=null)
        {
            m.put("domainId",this.domainId);
            if (this.capeId!=null) m.put("capeId",this.capeId);
            if (this.competencyId!=null) m.put("competencyId",this.competencyId);
            if (this.affectId!=null) m.put("affectId",this.affectId);
            if (this.motivationId!=null) m.put("motivationId",this.motivationId);
            if (this.metacogId!=null) m.put("metacogId",this.metacogId);
        }
        return "default";
    }

    public void setAffectId(String affectId) {this.affectId = affectId;}
    public void setCapeId(String capeId) {this.capeId = capeId;}
    public void setCompetencyId(String competencyId) {this.competencyId = competencyId;}
    public void setDomainId(String domainId) {this.domainId = domainId;}
    public void setMetacogId(String metacogId) {this.metacogId = metacogId;}
    public void setMotivationId(String motivationId) {this.motivationId = motivationId;}
}
```

Listing 3.3: The Velocity template used to generate the OLM view

```
<HTML>
<TITLE> Open Learner Model </TITLE>
<HEAD></HEAD>

<BODY topmargin="0" leftmargin="0" marginwidth="0" marginheight="0">
<P align=center>
<APPLET
   name="olmapplet"
   code="olmgui.OLMMainGUI.class"
   archive="../applet/olmgui.jar,../applet/xmlrpc-applet.jar,
            ../applet/concurrent.jar,../applet/touchgraph.jar"
   width="95%" height="95%" >
       <param name="olmCoreServiceUrl" value="${olmCoreServiceUrl}">
#if (!$domainId || $domainId.isEmpty())
    <!-- no initial belief descriptor -->
#else
    <param name="domainId" value="${domainId}">
  #if ($capeId && !$capeId.isEmpty())
    <param name="capeId" value="${capeId}">
  #end
  #if ($competencyId && !$competencyId.isEmpty())
    <param name="competencyId" value="${competencyId}">
  #end
  #if ($affectId && !$affectId.isEmpty())
    <param name="affectId" value="${affectId}">
  #end
  #if ($motivationId && !$motivationId.isEmpty())
    <param name="motivationId" value="${motivationId}">
  #end
  #if ($metacogId && !$metacogId.isEmpty())
    <param name="metacogId" value="${metacogId}">
  #end
#end
</APPLET>
</P>
</BODY>
</HTML>
```

3.3 The OLM-GUI

The OLM-GUI is basically an applet, embedded in a browser window and deployed when requested by the appropriate URL. The description of the applet itself will be given in the next section.

When the URL of the OLM is activated in the browser and interpreted by the OLM-Controller, the Velocity script (see listing 3.3) used to deploy the OLM-GUI is generated. It produces a very simple HTML document in which the applet is embedded. The parameters extracted from the URL and interpreted by the OLM-Controller (i.e. the URL of the OLM service and the initial descriptor – if any) are passed to the OLM by standard mechanism (i.e. with as many <PARAM> tags as needed).

3.4 The OLM-Core

The OLM-Core is the model of our MVC-compliant OLM. In practice, the OLM-Core acts as an XML-RPC server with which the OLM-GUI communicates as a client. This communication is channelled through a couple of "handlers" implementing the various tasks the OLM-Core provides the OLM-GUI with. The following list gives a description of the main handlers (internal methods and data structures are not described, please refer to the JavaDoc documentation).

- `public OLMQueryResult getUserInfo()`

  This handler is called by the OLM-GUI to retrieve information about the learner using the OLM. It includes the identifier, the full name, the language and the learning context, as extracted from the LeActiveMath user-profile.

  The language is the most important information retrieved here because it is used by the OLM to internationalise the various elements of its interface (see section 4.4).

- `public OLMQueryResult getJudgmentOn(Vector beliefId,boolean flat)`

This handler is called by the OLM-GUI when the learner has requested a judgement from the OLM on the given topic. The request is identified by the belief descriptor `beliefId`; the parameter `flat` determines whether the evidence has to be collected uniformly or as clusters, highlighting further diagnosis in the judgement (see section 2.2.3 for a discussion on re-clustering the evidence).

The handler returns a hashtable (i.e. a set of attribute/values) which will be used to build a data structure reflecting the Toulmin Argumentation Pattern applied for organising information in the OLM [4]. This data structure contains all information required by the OLM-GUI to present the judgement to the learner, organise its justification and control its challenge (see section 3.7 below for discussion on the synchronisation between the OLM-Core and the OLM-GUI).

- `public OLMQueryResult getConceptMap(String mapId)`

  This handler is called by the OLM-GUI during its initialisation to retrieve all the various topics maps used by the Learner Model to represent the different layers of its beliefs (i.e. domain, competency, affect, motivation, metacognition and CAPEs) and their elements.

  The map, given by its unique identifier `mapId`, is returned as a hashtable which contains the list of all the nodes (i.e. the elements – topics and associations – constituting the concept map, given with their unique identifier, their name and description), a list of all the edges between the nodes and the "top-level" node of the map (i.e. the node assumed to represent the map itself). This data structure will be used in the OLM-GUI to feed the various interface elements needing this information (such as the Descriptor View – see section 4.2.1.1 – or the Navigation View – see section 5.3).

- `public OLMQueryResult getNextDialogueMove()`

  This handler is called by the OLM-GUI whenever the learner is requesting help about what to do next. The OLM-Core keeps track of the past and current moves of the learner and is therefore able to re-contextualise this request and to make a "relevant" suggestion.

  The suggestions are always made in term of the next move the learner should do in order to keep going (see section 4.3 for a list and description of the dialogue move supported by the OLM). This suggested move can be further detailed by any relevant information (for example a suggestion using the `SHOWME` move will be extended by a list of possible belief descriptors that the learner may decide to explore; such a list is built on the basis of the unresolved issues stored by the Learner Model).

- `public boolean sendSuggestion(Vector beliefDesc,Hashtable params)`

  This handler is called when the OLM suggest to the learner to perform some exercise in order to try to resolve a disagreement between them. A request is therefore made to the Tutorial Component to build a new sequence of exercises (see section 3.6 below), based on the belief descriptor `beliefDesc` and the source of disagreement, as identified by `params`, which will contains information such as the target of the disagreement (e.g. the claim, the evidence X, the attribute Y of exercise Z, etc.).

- `public boolean sendDialogueMove(Hashtable params)`

  This handler is called everytime a dialogue move is made by the OLM or the learner in order to store this event – for further use by the OLM– and to communicate any information that the Learner Model needs in order to update its beliefs.

[4] This reconstruction is necessary because of the `XML-RPC` protocol used to communicate between the OLM-Core and the OLM-GUI. The protocol only accepts a couple of basic types and structures such as the attribute/value or vector. Any complex data structure has to be "converted" into a combination of `XML-RPC` compliant types before being sent to the GUI and reconstructed in the GUI for easing its manipulation. See http://ws.apache.org/xmlrpc/ for a description of the `XML-RPC` implementation in Java.

3.5 Communications with LeActiveMath

As mentioned above, the communication inside the OLM (i.e. between the OLM-Core and the OLM-GUI) is ensured by the `XML-RPC` protocol. Communication between the OLM and LeActiveMath is ensured by the event framework supplied by LeActiveMath (see Deliverable D8 [10]) and remains mostly located within the xLM.

The front-end of the xLM is the `XLMEventManager`, whose job is to receive the event published by LeActiveMath and to dispatch them into the various sub-components of the xLM. It also act, in a similar way, for the events generated by the sub-components. This is the mechanism used for the OLM to communicate with the xLM.

Three different events are generated by the OLM, to be intercepted and interpreted by the Learner Model:

- `OLMMetacogEvent` is published when evidence of the metacognitive abilities of the learner are detected through their interaction with the OLM. This diagnosis is made on the basis of the dialogue moves performed by the learner, taking into account their nature (e.g. for exploration purpose like `SHOWME` or for challenge purpose like `DISAGREE`), repetition and context to refine the evidence.
- `OLMChallengeEvent` is published by the OLM every time a "challenge" on its judgement is made by the learner. Here, "challenge" has to be taken in a broad sense, including all situations where the learner agrees with the judgement, where the learner disagrees with it and where the learner decides to "move on" and refuses to commit himself. The event provides the Learner Model with a new piece of evidence about the learner's ability on the challenged topic. The nature of the evidence – and how it will be taken into account for updating the belief – depends on how (agreement, disagreement, avoidance) and where (on the claim or on one of the warrant/backing justifying it) the challenge was performed by the learner.
- `OLMMoveEvent` is published by the OLM every time a dialogue move is made, either by the learner or by the OLM.

Tables 3.1, 3.2 and 3.3 describe the attributes defining both events.

`String moveID`	The identifier of the dialogue move
`Vector descriptor`	The belief descriptor, target for the inference of the learner's metacognition
`double depth`	An estimation of the depth of the monitoring/control on this belief (should be normalised, so with a value between 0 and 1)
`int initiative`	Indicates if the move has been made on the learner's own initiative (1) or on the OLM's suggestion (0)

Table 3.1: Attributes of the OLMMetacogEvent

3.6 Integration with LeActiveMath Front-end

As mentioned above, the bulk of the communication with LeActiveMath will take place within the boundaries of the xLM– essentially with the Learner Model– using the event framework. Nevertheless, there are two situations where the OLM and LeActiveMath are cooperating directly: the deployment of the OLM from the front-end and the suggestion made by the OLM to the Tutorial Component for proposing further content to present to the learner.

`String moveID`	The identifier of the dialogue move
`Vector descriptor`	The belief descriptor, target of the learner's challenge
`String target`	The target of the challenge, i.e. one of the Toulmin's element (claim, backing, etc.)
`double confidence`	Contains the confidence of the learner's challenge (between 0.1 and 0.9, since we assume the learner to be neither totally confident nor totally uncertain)
`double intransigence`	Contains the degree of intransigence of the learner, ie how much he/she is prepared to compromise with the OLM
`double level`	If the target is CLAIM, contains the level that the learner thinks represents his/her true abilities
`int evidence`	If the target is WARRANT or BACKING, it contains the index of the evidence (in the belief) that has been challenged. Otherwise, it contains -1
`String attribute`	If the target is BACKING, it contains the attribute of the event that has been challenged (eg "difficulty" of an exercise, "pride" from the SRT, etc)
`String value`	If the target is BACKING, it contains the value of the attribute that the learner thinks should be taken into account by the LM (eg "very_easy" for "difficulty").

Table 3.2: Attributes of the OLMChallengeEvent

`String moveID`	The identifier of the dialogue move
`boolean isOLM`	Indicates if the move has been played by the OLM (`true`) or by the learner (`false`).
`Vector descriptor`	The belief descriptor, target of the dialogue move.
`String target`	The target of the move, i.e. one of the Toulmin's element (claim, backing, etc.) or `null` if not applicable.

Table 3.3: Attributes of the OLMMoveEvent

3.6.1 Deployment of the OLM

The Maverick-Velocity approach used in structuring and implementing the OLM makes it very easy to implement its deployment at the front-end level, even ensuring a mixed-initiative strategy, where both the learners and LeActiveMath can request this deployment.

The OLM is uniformly accessed by its URL, as specified above. It is only on activation that instances of the OLM-GUI and OLM-Core will be created for each learner individually (and in a transparent way). This means that the OLM can be deployed both manually by the learner (see for example figure 3.2 showing the main menu of LeActiveMath with a direct shortcut for the OLM) and programmatically by the system, using a single explicit URL.

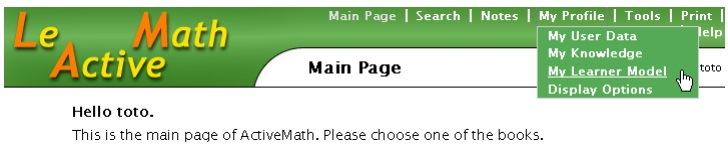


Figure 3.2: Snapshot of LeActiveMath main menu, containing a shortcut to the OLM

The Tutorial Component for example is already using such possibility when implementing the tutorial strategies (see Deliverable D20 [16]). One of the tasks of the strategies can be introduced to explicitly request learners to access their model (for example at the end of the sequence). This

task, besides introductory texts and prompts, contains a shortcut to the OLM. Depending on the circumstance, parameters can be added to the plain URL to specify the topic with which the OLM should be first deployed (see figure 3.3).

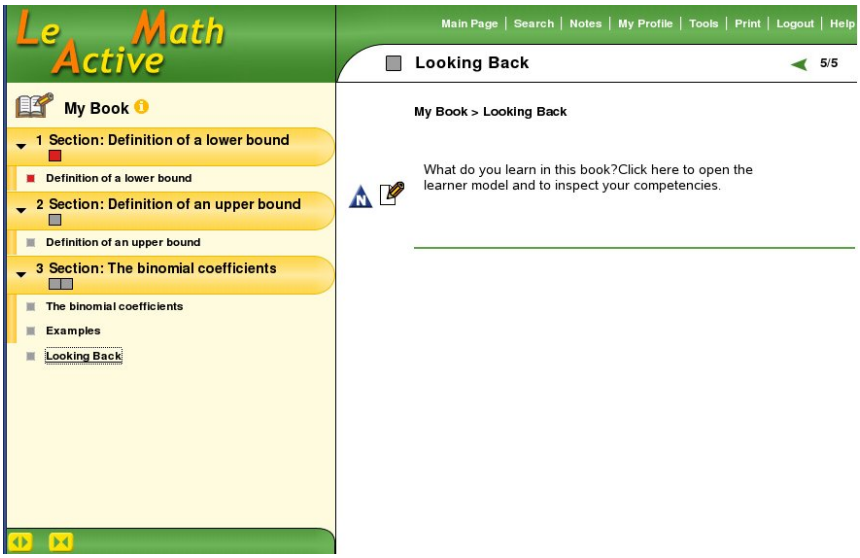


Figure 3.3: Snapshot of a learning object containing a reflection task, used to deploy the OLM

3.6.2 Suggestions to the Tutorial Component

When the negotiation between the OLM and the learner on a topic reaches a point where no agreements could be made, the OLM has the possibility to suggest to the learner to perform more exercises before resuming the discussion (the assumption being that more evidence gathered by the Learner Model may lead either the OLM or the learner to change their position on the disagreement).

Such a possibility is offered by the Tutorial Component – with its mechanism for selecting pieces of content (see Deliverable D20 [16]) – and is providing us with an interesting bridge between the tutorial and self-reflective aspect of LeActiveMath.

This functionality has been formally agreed between WP4 (OLM) and WP3 (Tutorial Component) and is in its early stage of implementation (most of the remaining investigations are focusing on which criteria are using to select the extra exercises, how they are presented to the learner and how they are referred to by the Learner Model).

3.7 Synchronisation between the LM and the OLM

LeActiveMath, being a web-based learning environment, is by nature a free-discovery environment where very few possibilities for "restricting" the learner actions or sequences of interaction can be made. The OLM does not escape this situation, as its deployment as a browser-embedded applet does not prevent simultaneous access to the rest of LeActiveMath. Therefore we have to be aware of potential problems resulting of the (de-)synchronisation between the OLM and the Learner Model.

Let's envisage a situation where the learner, after doing a couple of exercises in LeActiveMath, accesses the OLM and requests a judgement on the topic he has just explored. The OLM consults

the Learner Model and retrieves the current belief associated with the topic, ready to justify it at the learner's request. But before doing precisely this, the learner – who still has access to LeActiveMath front-end – moves the OLM to the background, and performs another couple of exercises.

At that stage, the Learner Model is intercepting and interpreting the events generated by the new exercises and, therefore, updates the related beliefs – including the one that has just been displayed in the OLM. Therefore the belief stored in the Learner Model and displayed in the OLM are not any more synchronised, without the learner's knowledge. When the request for justification is made, the evidence collected and presented to the learner as a result is in fact representative of the new state of the belief and not of the belief currently displayed. And since it is possible for both beliefs to be significantly different, this is an issue that we cannot ignore.

The first possibility would be for the OLM to be signaled of any changes in the Learner Model, in particular about the belief currently presented. If it is technically possible, by using the event framework supplied by LeActiveMath, it nevertheless introduces challenging behaviour at the interface: how do we tell the learner that the belief has now changed and need to be updated? How do we make this update explicit at the interface level or what do we do if we don't?

The second approach – which is currently implemented in the OLM– consists in postponing any synchronisation and update at a stage in the learner interaction with the OLM where it will not introduce any problem of notification and feedback. The nature and organisation of the dialogue framework supported by the OLM (see section 4.3) makes this approach possible. When the learner requests a judgement on a topic, all the information required for displaying this judgements, but also for supporting its further investigation and challenge, are immediately uploaded to the OLM-GUI. It means that the dialogue will be made on the basis of this set of information and without further request to the OLM-Core and the Learner Model. Even if the learner does perform some exercises at the same time as exploring a judgement, the dialogue will remain consistent with the judgement displayed at the first place. It is only when the learner requests the same judgement again that the updated belief will be taken into account.

4 Implementation

In this section, we will describe the implementation of the visible part of the OLM, i.e. its Graphical User Interface.

4.1 The General GUI

The main GUI of the OLM is built around four distinct parts (see figure 4.1):

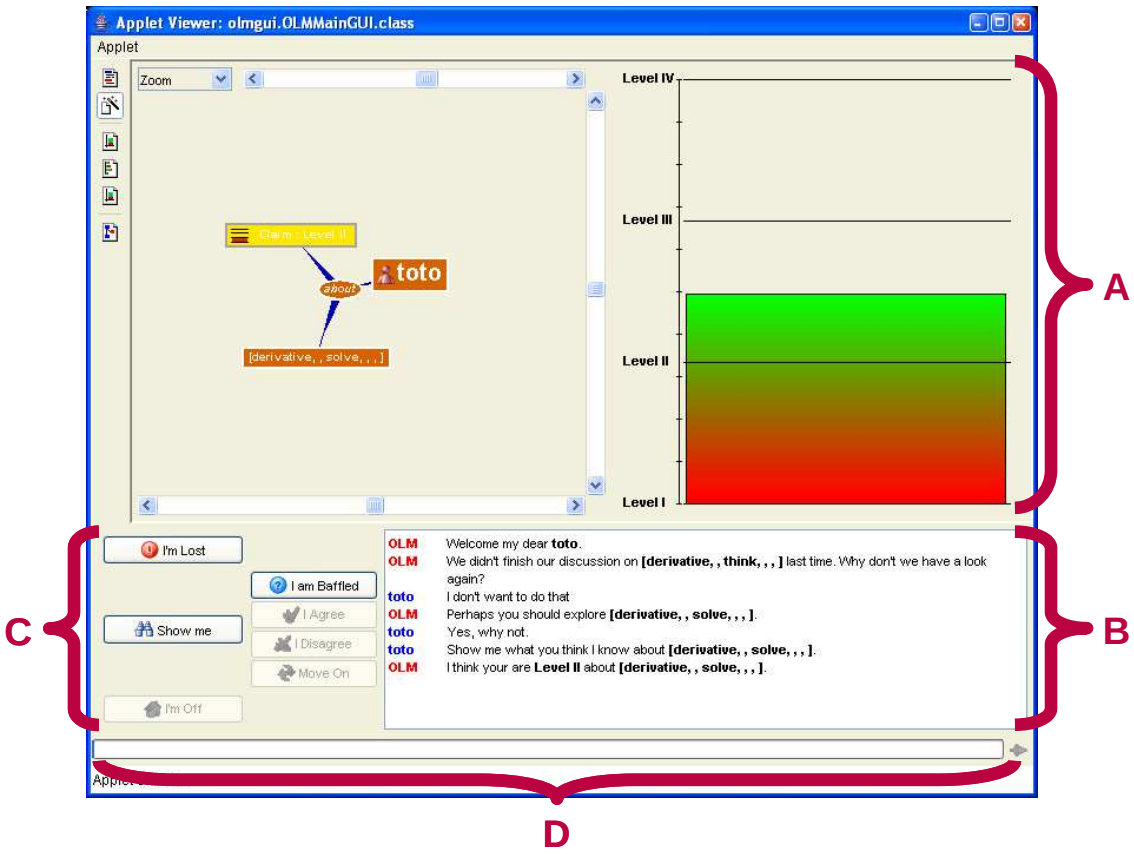


Figure 4.1: A snapshot of the OLM Graphical User Interface.

A the top half of the GUI contains the main source of interaction between the learners and the OLM. The nature and exact content of every external representation is described in section 4.2.

B The bottom right-hand side of the main applet contains a scrolling text panel used to provide the learners with a natural-language transcription of their interactions with the OLM. This panel has two purposes: first maintaining a log – in a readable format – of the history of the learners' interactions and, second, complementing the external representations displayed in the upper-part of the OLM with some textual explanations about what is going on. The content of this external representation is described in section 4.4.

C The bottom left-hand side of the main GUI contains the dialogue panel, i.e. a couple of widgets used by the learners to communicate with the OLM. They represent the basic dialogue moves that the learners could perform in a given situation: "I'm Lost" to request help and support for the current situation, "Show Me" to request the OLM to display a given belief, "I Disagree" to challenge the OLM decisions, etc. A complete description of the dialogue moves, both learners' and OLM's, is given in section 4.3.

D Finally, the bottom end of the GUI contains a small status bar used to convey interface-related information to the learners[1]. It displays short textual messages related to the current state of the OLM such as tooltips for the selected widgets, error messages issued by the XML-RPC communications, etc.

4.2 External Representations

The external representations used in the OLM can be divided into three categories, representing the three stages of the dialogue between the learners and the OLM.

- The first stage is the exploration of the content of the Learner Model, i.e. the navigation through the various belief held in the system and the selection of a particular topic of discussion.
- The second stage is the justification by the OLM of the decision made on the topic selected by the learners.
- The last stage is the challenge by the learners of some elements of the argument presented by the OLM.

Obviously, none of these stages are imposed on the learners but left at their own decision (within the limit of the mixed-initiative supported by the OLM, see the `STARTUP` dialogue move in section 4.3). Nonetheless, they have to be operated in this particular order, i.e. a belief has to be selected for justification before the learners could challenge the OLM's decisions.

4.2.1 Exploring the content of the Learner Model

4.2.1.1 The Descriptor View

The current implementation[2] of the Descriptor view applies a very straightforward interface (see figure 4.2), i.e. the explicit building of a belief descriptor by the learner.

The right-hand side of the view contains a couple of placeholders arranged in a way to reflect the organisation and dependency of the layers in the Learner Model, following the principles established in section 2.1.1: the Domain topic at the bottom, supporting on one hand the Competency topic, the Affect and Motivation topics and finally the Metacognition[3] topic, on the other hand the CAPEs.

The intuitive idea is for the learners to pick up the topics of their interest and manually build a descriptor to submit to the OLM. To do that, the left-hand side of the view contains a couple of tabbed lists, each of them representing one of the layer of the Learner Model (Domain,

[1] This should not be confused with content-related information and feedbacks, as provided by the various external representations used by the OLM.

[2] See section 5.3 for a discussion about an alternative – but more demanding – interface for the navigation through the content of the OLM.

[3] See section 5.4 for a discussion on hiding information.

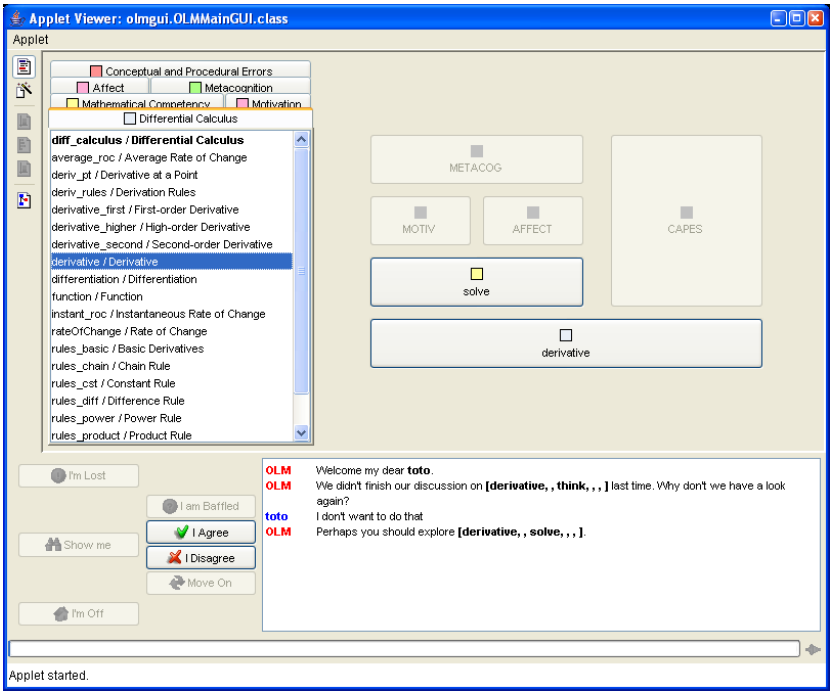


Figure 4.2: The Descriptor View used to specify the belief to explore in the OLM.

Competency, etc.) and containing all the topics available for the learners to explore. By simply dragging one topic from a list and dropping it into the proper placeholder[4], the descriptor is built step-by-step.

Once the learner is happy with the descriptor they want to explore, a simple click on the SHOWME button will prompt the OLM to present its judgement – if any – on the topic.

A couple of points are worth mentioning about the Descriptor view:

- As mentioned previously, there are constraints on how a belief descriptor is structured (no CAPEs and Competency together, a domain topic always present, etc.). It is still not clear what is the best way to make the learner aware of these constraints: either by preventing any wrong configuration before or indicating such problems after validation. In the current implementation, we opted for the second approach, as it improve the feeling of a "real dialogue" between the learner and the OLM. In the case of a malformed configuration, the OLM will make a statement such as *"Sorry, I don't understand your question"* and will, on demand by the learner, provide information about why the question was badly formulated.
- At this stage of the dialogue, the LOST move takes a more pro-active role when used. Instead of just issuing a generic hint or support, the OLM asks the Learner Model for any unresolved issues with the learner and then suggests the learner to consult the belief again. See below for more information on the LOST move.

4.2.1.2 The Argument View

The presentation and justification, step-by-step, of the belief is controlled by the Argument view (see figure 4.3). The purpose of the View is twofold: to provide the learner with a representation

[4]Well, in fact the operation is done by double-clicking rather than drag-and-drop!

of the logic of the justification of the judgement made by the OLM and to act as an interface to navigate between the various external representations made available to provide more details about the different components of the belief.

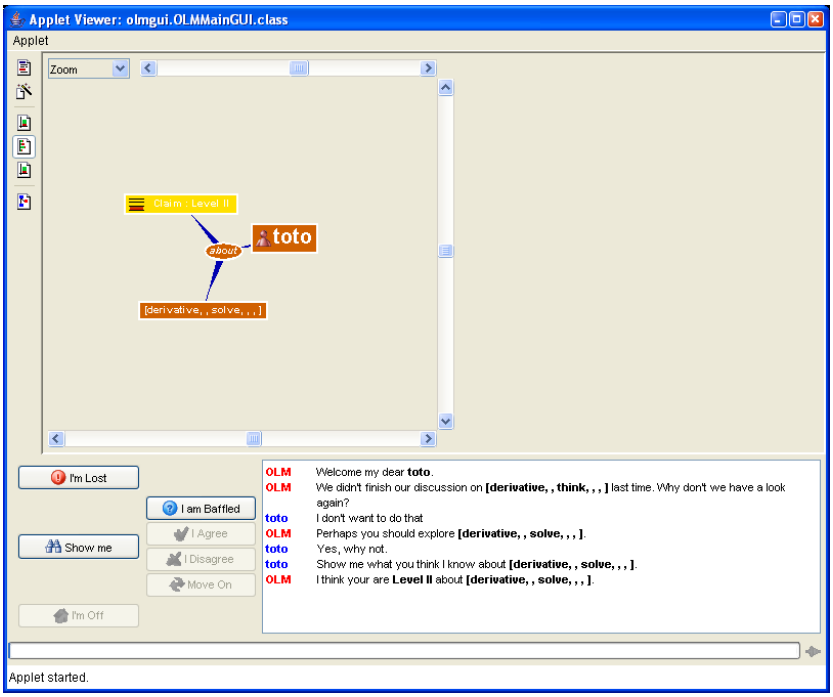


Figure 4.3: The Argument View, displaying the structure of the various elements of the argumentations.

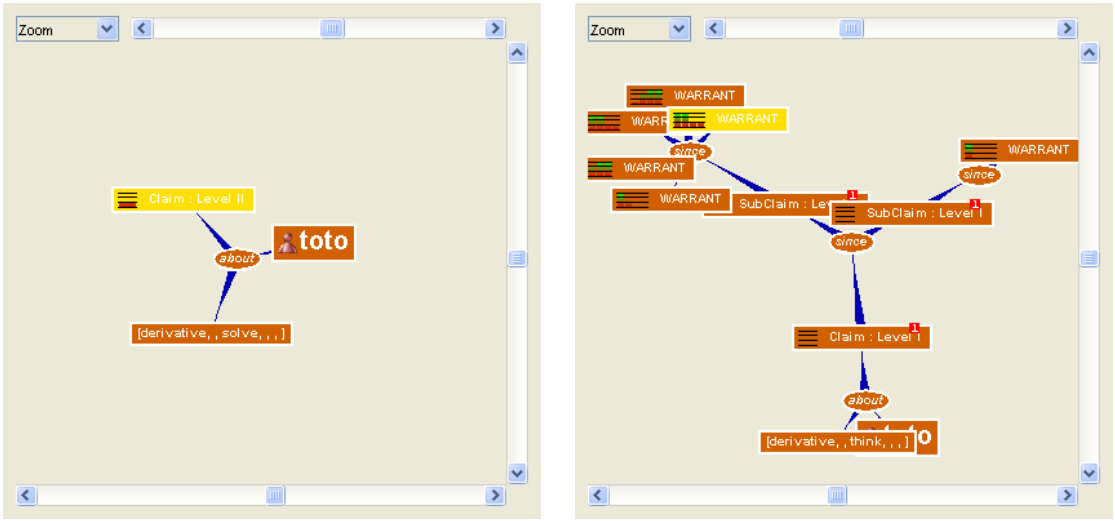
As mentioned in section 2.2.1, beliefs about the learner are organised by a Toulmin-inspired argumentation pattern which is virtually represented as such in this view, under the appearance of a dynamic and interactive graph.

Each of the nodes of the graph is associated with one of the elements of the argumentation pattern: the claim node associated with the summary belief, the data node associated with the belief itself, the warrant and backing nodes associated with individual evidence, etc. The shapes, colours, labels and icons of the nodes are appropriately designed in order to provide a quick and unambiguous identification of the corresponding element.

These nodes are also reactive to the learner's interaction and act as a shortcut to access the appropriate external representation, which will be immediately displayed beside the Argument view. Thus, selecting the claim node – or a sub-claim , if any – will open the Claim view and update its content accordingly, selecting one of the evidence nodes will open the Warrant/Backing view and update its content based on the selected evidence, etc. (see for example the figure 4.5 showing both the Argument and Claim view after the learner selected the claim node in the graph).

Some intermediary nodes, not reactive to learner's interaction, have been added to provide meaningful associations between the important parts of the graph; they are mostly linguistic add-ons for improving both the readability and the layout of the graph.

The content of the Argument view will be expanded on demand, following the learner's request for further justification about the OLM judgement on the descriptor (by using the BAFFLED dialogue move). Figure 4.4 shows two stages of the expansion of the argument pattern. On the left, the argument is presented as when the learner initially requested a judgement about the selected belief: the target of the request, as well as the claim made by the OLM are the only visible nodes



(a) The Argument structure initially deployed

(b) The Argument structure expanded down to the evidence

Figure 4.4: Two snapshots of the Argument View, at different stages of the argumentation.

on the graph. On the right, the argument has been further expanded by the learner requesting more justification for the judgement. Two sub-claims (see section 2.2.3) can be seen on the graph, as well as the list of evidence (represented by the WARRANT nodes) supporting one of them.

4.2.1.3 The Claim View

The claim view is the main view for the belief. It is automatically displayed when the learners build a descriptor and query the OLM about it. Its main purpose is to provide learners with a succinct overview of their overall ability on the requested topic.

Two complementing pieces of information are simultaneously conveyed with this external representation (see figure 4.5): the discrete and the continuous values of the summary belief. The discrete value corresponds to the level proper and is given in the dialogue view, in the claim node on the graph and in the major gridlines in the bar chart. As mentioned before, the "placeholders" `Level I`, `Level II`, etc. are mapped to the relevant phrasebook associated with the dominant layer of the descriptor (see section 4.4). The continuous value of the summary belief is given by the bar chart itself, it's size and proximity to the gridlines reflecting both the dominant level, the tendency of the learner's ability (i.e. `Level II`but on the upper quadrant) and the conviction of the OLM on its judgement (the closer the bar is to a gridline, the more certain the OLM is about its own judgement).

4.2.2 Justifying the content of the Learner Model

Once a belief has been selected and initially displayed in the OLM, the learners can now express their puzzlement or interrogation about the reasons why the OLM believes this about them. This is basically done by using the `BAFFLED` dialogue move on one of the node in the claim view. Such a move will provoke the expansion of the argumentation pattern into further details: when baffled about the claim (i.e. summary belief), the data (i.e. the belief) will be expanded; when baffled about the data, the warrant (i.e. the evidence) will be expanded[5]; etc.

[5]Or the sub-claims if any, see section 2.2.3

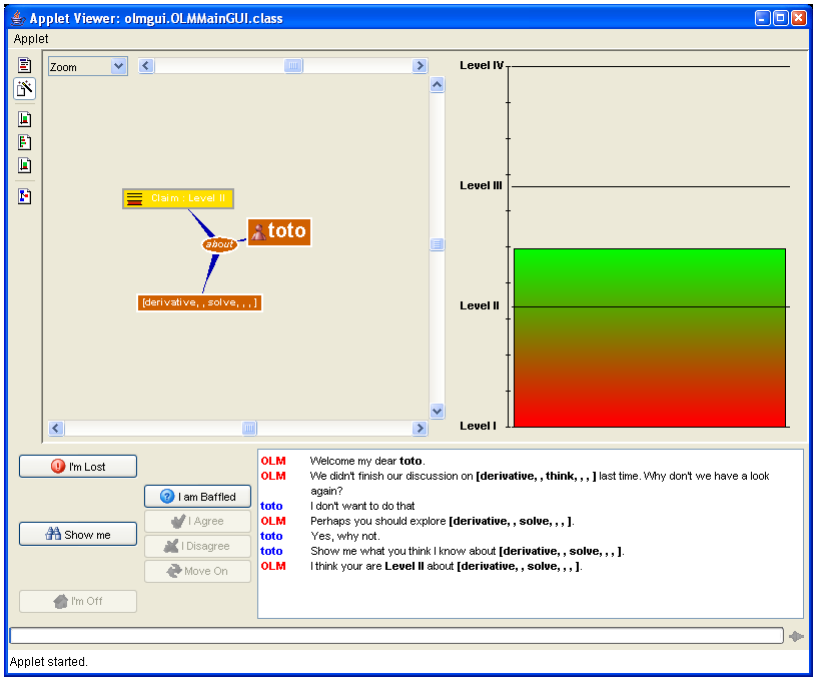


Figure 4.5: The Claim view displaying the summary belief

Two views are available for the learner to explore such extra information: the Data View and the Warrant/Backing View.

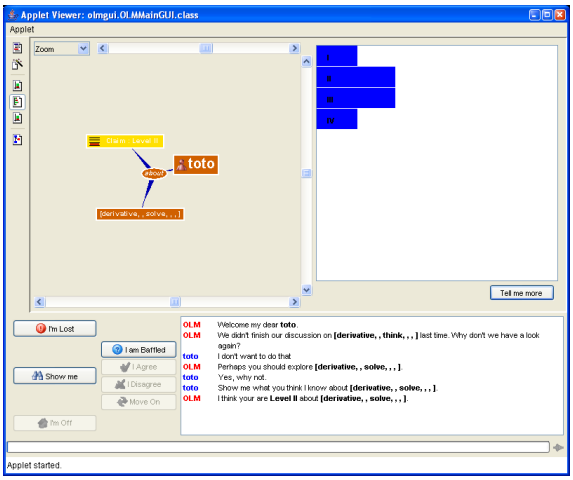
4.2.2.1 The Data View

The Data View (figure 4.6) is used by the OLM to support the exploration and analysis of the data part of the Toulmin Argument Pattern, i.e. the belief itself held by the Learner Model. The view itself contains three different types of external representations (pignistic, certainty and mass distributions), each of them detailing a bit further the meaning of the belief. A widget, labelled `Tell Me More` and situated just below the view, provides the learner with a contextualised BAFFLED dialogue move[6] used to switch between the three representations.

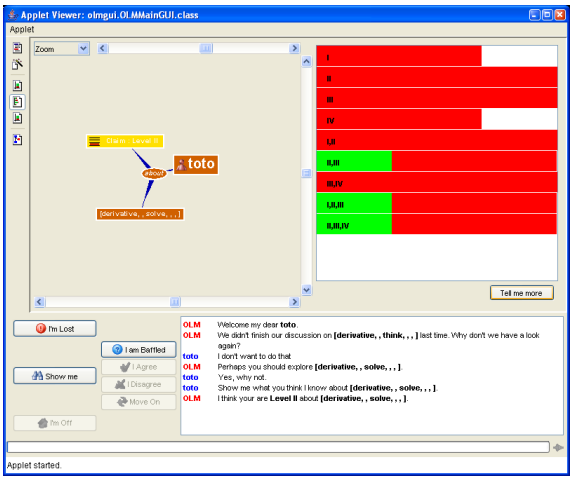
The Pignistic External Representation (figure 4.6(a)) is a natural extension of the summary belief, by displaying the normalised distribution on the four singletons (`Level I`, `Level II`, `Level III`and `Level IV`). It's purpose is to provides the learner with a first shallow justification for the summary belief stated by the OLM. In essence, it's message is *"Well, I said you are `Level N`because `Level N`is the most likely conclusion of my interpretation of your behaviour"*.

The Certainty External Representation (figure 4.6(b))highlights how the distribution on the singletons has been generated, by displaying the certainty (in green) and the plausibility of the

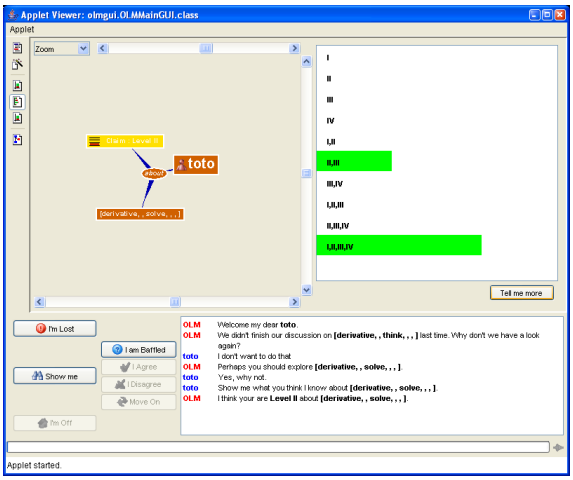
[6]This is one of the few cases where the widgets used to convey the dialogue moves are not located in the main dialogue panel. This was initially for implementation purpose,i.e. easing the detection of the context on which the dialogue move is fired and not overloading the main button on the dialogue pane. It remains an issue to decide if the interface should remain consistent or if local "adaptation" will facilitate the usability of the OLM. Whatever its location, the widget is nevertheless implementing a BAFFLED move – see section 4.3 – so that the response of the OLM remains consistent and the behaviour of the learner could be still analysed, in particular for the metacognition diagnosis.



(a) The Pignistic ER



(b) The Certainty ER



(c) The Mass Distribution ER

Figure 4.6: The Data view and its different external representations

learner's diagnosed ability on every possible set: the singleton (`Level I`, etc.), doubletons (`Level I-II`, etc.), and tripletons (`Level I-III`, etc.). The emphasis of this external representation is less on the levels per se than on the set: the "wider" the sets are (tripletons for example), the less convinced the OLM is about the learners' ability, in the sense that it prefers to make a wide guess rather than "taking the risk" of narrowing its judgement (toward a singleton). The Certainty external representation is therefore a vehicle for the learner to grasp the tension between the certainty of a decision and its focus.

In order to facilitate such reading of the external representation, the sets can be all displayed at once, or gradually by starting with the tripletons, then the doubletons and finally the singletons. Coupled with the pignistic external representation, this gradual expansion of the belief's internal structure supports a (simple) form of scaffolding.

The Mass Distribution External Representation (figure 4.6(c)) is the last of the external representation of the belief itself and is simply a distribution of the total amount of information extracted from the evidence across the sets. Once ranked by importance, this representation provides the learner with insights about how the various evidence received by the Learner Model to build the model directly supports the various assumptions made on the overall learner's ability.

But it does not explain how individual evidence had such an impact. To be able to access this level of description, the learners has to keep asking for further explanation about the judgement made by the OLM. By using the `BAFFLED` dialogue move on the Data node of the Argument view, the argumentation graph is now expanded such that every piece of evidence supporting the claim are now displayed. Selecting one opens the Warrant/Backing View

4.2.2.2 The Warrant/Backing View

The Warrant/Backing View (figure 4.7) is a dual external representation intended to display the evidence justifying the belief.

On the top part of the view lies the mass distribution representing the numerical interpretation of the evidence. Similar in its reading to the mass distribution view of the belief, it put an emphasis on the sets that are the most likely to be the explanation of the "performance"[7] of the learner, as diagnosed during the event that generated this evidence.

On the bottom part of the view, the qualitative and quantitative information that categorise the event is presented to the learner. The number, nature and content of these parameters will vary a lot from one piece of evidence to another one: success rate, difficulty and competency level for an exercise, the source of a propagation of evidence and its relation with the target, etc. They are currently displayed as a table of attributes and values.

Whatever the nature of the events, they will all be associated with one of the learning object of LeActiveMath (e.g. the exercise that has been achieved by the learner, the exercise at the end of which the learner expressed his satisfaction, etc.). The learner therefore has always the possibility to display this object again in a dedicated LeActiveMath browser, on request by using the "Show Item" button on the view).

It has to be noted nevertheless that, to the best of our knowledge, LeActiveMath does not support any playback functionality for replaying past interactions of learners. Even if the LH will store all elements of the interactions (steps made by learners, all output produced, all hints provided by the system, every single assessments of the learners' performance on the exercise, etc.) it is not possible to regenerate it. The "Show Item" button will only open the learning object as

[7] The term "performance" is here to be taken in a broad sense, as the underlying event could be of many different types: exercise finished by the learner, self-report about there affective state, propagation of evidence from a close topic, etc.

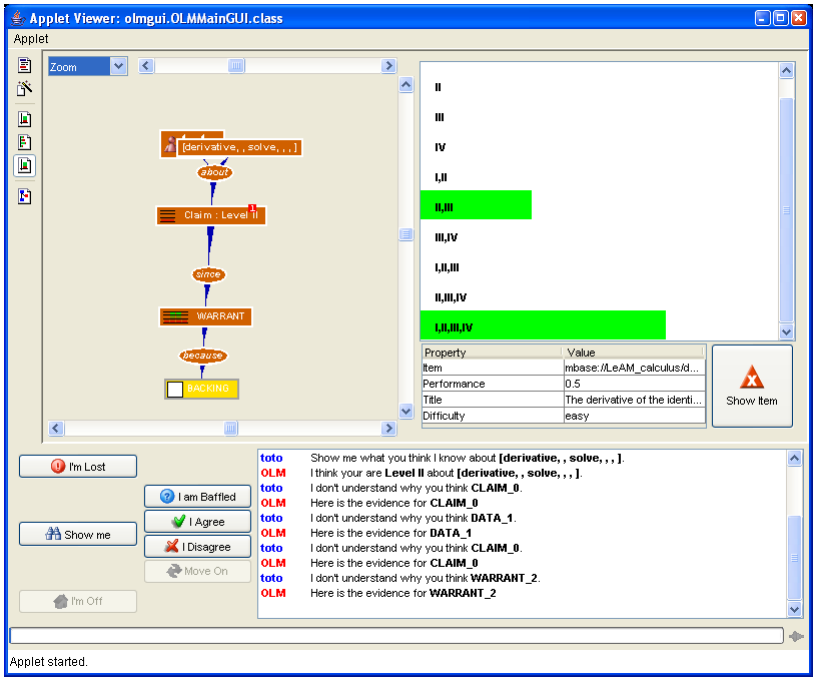


Figure 4.7: The Warrant/Backing view, displaying the Mass Distribution generated by the corresponding event

presented by the dictionary of LeActiveMath (see Deliverable D15 [15]). Recontextualising the event therefore relies on the description of the exercise (*"Do you remember doing this exercise?"*) rather than on its execution (*"Do you remember when you answered* $sin(x^2)$ *on this step?"*).

4.2.3 Challenging the content of the Learner Model

At any time during the exploration of a belief and its justification the learner could challenge the OLM on its judgement. As mentioned in section 2.2.2, there are three elements of the argumentation pattern that the learner could challenge: the claim, one of the pieces of evidence and one of the attributes of one of the events. Since they requires different tasks and have different impacts in the Learner Model, each of these challenges has its own interface for the learner to perform it. All of them are organised within the Challenge View.

4.2.3.1 The Challenge View

By using the DISAGREE dialogue move on one of the node of the argument graph (claim, warrant and backing), the learner states his disagreement with the relevant information. The GUI now opens the Challenge view for the appropriate context (see figure 4.8 and the interface for challenging a claim) and prevent any other free navigation in the OLM until the disagreement has been resolved, successfully or not.

At that stage, a challenge is recorded by the OLM, wether the learner pursues it up to its obvious end by validating his alternative statement or by finally giving up and cancelling it. In both situations, an event is sent to the Learner Model, indicating in the first case what kind of change has to be taken into account and, in the second case, that the corresponding belief has to be marked as an "unresolved issue".

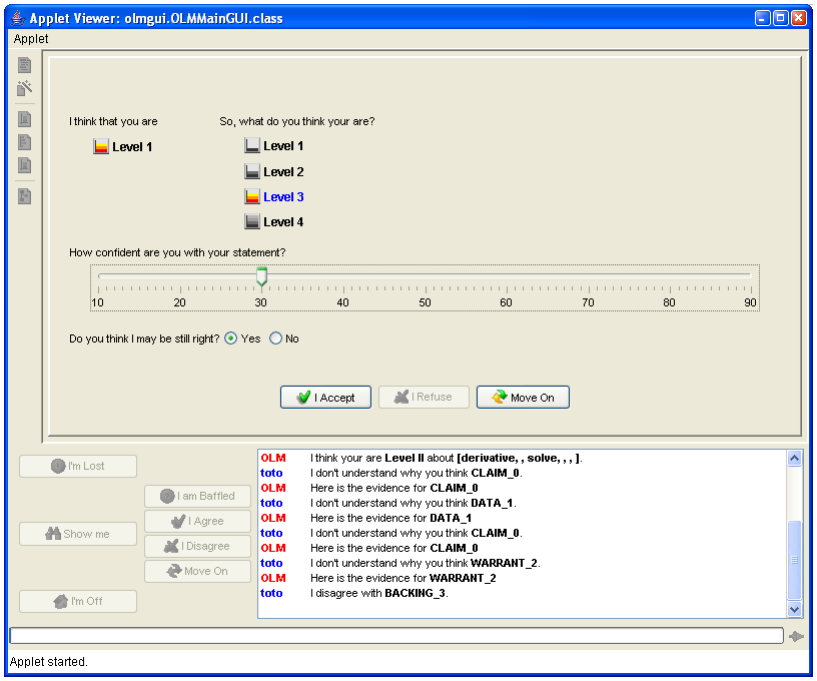


Figure 4.8: The Challenge view, with the interface allowing the learner to challenge the claim made by the OLM

The interface for challenging the OLM judgements currently supports a shallow negotiation. The learners are expected to provide the OLM with their own qualification of their disagreement (by stating their own confidence in their alternative claim and their own intransigence with respect to the conflict). The policy hence implemented means that the OLM always gives up in front of the learners (i.e. they can always state an alternative judgement).

4.3 Dialogue and Dialogue Moves

Learner's dialogue moves (e.g. BAFFLED, DISAGREE, LOST, etc.) are basically fired by using the proper button on the GUI. The move is immediately followed by an OLM dialogue move (e.g. HEREIS, WINDUP, UNRAVEL, etc.) that interprets the learner's move (using the context when needed), prepare the response to the request and displays it. At that stage, the interface is again ready for the learner to use one of the dialogue moves. From a Finite-State Machine point of view, it roughly means that the learner's moves are transitions whereas the OLM's moves are states.

Figure 4.9 represents all the dialogue moves currently implemented in the OLM, as well as their overall organisation. As documented in this chapter, the dialogue moves are basically organised into three groups. The first supports the *exploration* of the OLM and the belief it holds, by running series of SHOWME-HEREIS moves (i.e. by building a belief descriptor and showing the corresponding claim). The second supports the *justification* of a claim made by the OLM, by running series of BAFFLED-HEREIS moves (i.e. expanding the argumentation graph and showing the relevant external representation). The final group implements the *challenge* proper, by organising the negotiation on the selected topic and its outcome (either by an agreement between the OLM and the learner – AGREE or DISAGREE– or by one of them giving up the discussion – MOVEON or LETMOVE).

It has to be noted that, once in the justification part of the dialogue, the learners cannot go back to the exploration of the OLM until the challenge has been resolved one way or another.

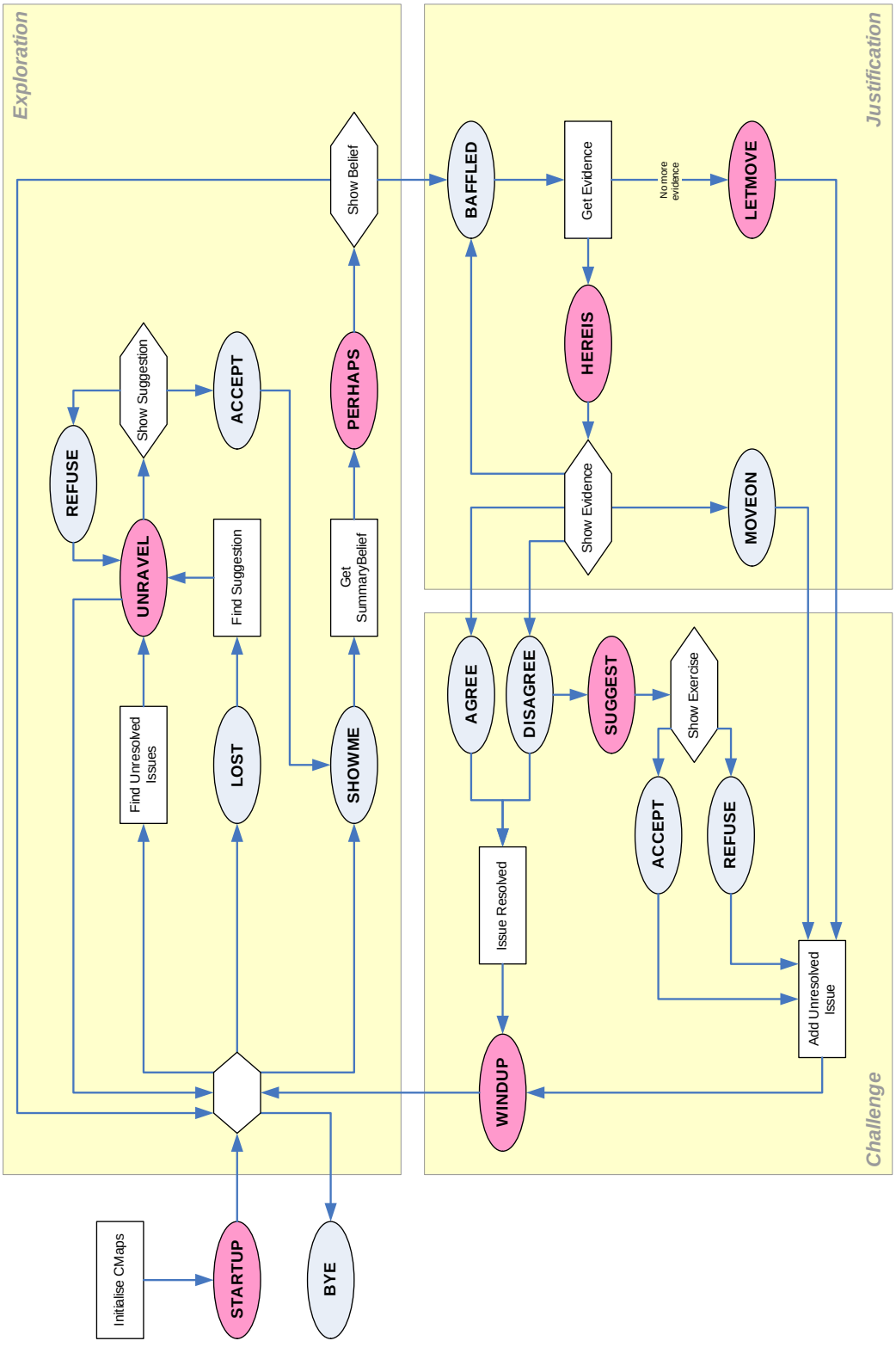


Figure 4.9: Overview of the organisation of the various Dialogue Moves in the OLM

The table 4.1 gives a list of all the dialogue moves currently implemented in the OLM with a description of their requirements and outcomes, as well as the list of the possible follow-ups.

Move	Description	Follow-ups
STARTUP	This is the starting point of any dialogue. Used by the OLM to initialise the GUI (in particular by uploading the topic maps from the OLM-Core, see 3.6), welcome the learner and initiate the dialogue, in a mixed-initiative manner. If the OLM has been activated by LeActiveMath, it checks if a topic is given as a parameter; if activated at learner's request, it checks if there are any outstanding issues remaining from previous interaction. In these cases, the OLM takes the initiative and suggests that the learner explore the given topic (by running the UNRAVEL move). If none of the above apply, the OLM lets the learner take this initiative and waits for a learner's move.	UNRAVEL LOST SHOWME BYE
BYE	Used by the learner to signify the end of the interaction with the OLM. The OLM being an applet embedded in a browser window, there is nothing we can do to detect when the learner manually close the window – and therefore the OLM. As the fact that the learner decided to stop using the OLM, this move is a (weak) way to enforce an explicit statement about shutting down the OLM.	-
LOST	Used by the learner to request help about the OLM, mostly in terms of a suggestion for the next move. The move is contextualised with the current activated move (one of the OLM's) to decide what kind of help is expected from the learner. Note that the expected help will always be in terms of further action (i.e. next dialogue move to perform); helps and hints about the interface, meaning of a widgets, etc. are not supported by the LOST move but by the status bar (see section 4.1).	UNRAVEL
UNRAVEL	Used by the OLM to react to a LOST move and suggests to the learner a move to perform. The nature of the suggestion depends on the context the LOST move took place. • If the request took place in the exploration phase of the dialogue (following a PERHAPS move), then the OLM checks, as during the STARTUP move, for any unresolved issues still known by the OLM. The suggestions (*"Perhaps you could have a look at topic X"*) are presented one by one to the learner who can reject (using the REFUSE move) or accept it (using the ACCEPT move). In the former case, the UNRAVEL move is reiterated with the next unresolved issue. In the latter case, the corresponding belief is presented to the learner by calling the SHOWME move. • If the request took place in the challenge phase (following a HEREIS move), then the OLM suggests exploring the argument deeper (e.g. explore the belief, explore the evidence, etc.).	LOST SHOWME BYE

(continued on next page)

Table 4.1: Description of the Dialogue Moves

(continued from previous page)

Move	Description	Follow-ups
SHOWME	Used by the learner to request any information the OLM holds about the learner's ability. This move is initiated only when the learner has built a (not necessarily correct) belief descriptor in the Descriptor View and has used the proper button in the dialogue pane. It can also be programmatically used by the OLM to display its judgement when the learner accepted to go ahead with one of the unresolved issues suggested in the UNRAVEL move.	PERHAPS
PERHAPS	Used by the OLM to state its judgement on a topic requested by the learner with the SHOWME move. The OLM uploads from the OLM-Core all the information required for populating the various external representations that will be used to explorer the belief, initialise the argumentation pattern and the Claim View.	BAFFLED LOST SHOWME BYE
BAFFLED	Used by the learners to request further information about how the OLM reached its judgement stated by the PERHAPS move. This move is dependent on the target selected in the argumentation pattern (i.e. claim, data, warrant and backing). The selected target is then explained by the OLM by running the appropriate HEREIS move. If the learners reach the "end" of the argumentation (i.e. the OLM has no further justification to support his claim) or keep repeating a BAFFLED move on the same target for too long, the OLM may decide to cut the argument short by running a LETMOVE move. As mentioned before, the BAFFLED move initiates the challenge part of the dialogue between the learner and the OLM. This move prevents any further request for a SHOWME move the challenge has been (successfully or not) resolved.	HEREIS LETMOVE
HEREIS	Used by the OLM to justify its judgement on the topic selected in the previous BAFFLED move. This justification is done by expanding the next node – if any – in the argumentation pattern and by displaying the corresponding view. If a node has already been justified (expanded), the OLM merely restates the reasons and switches to the relevant views.	BAFFLED AGREE DISAGREE MOVEON
MOVEON	Used by the learner to give up the negotiation on a judgement. MOVEON is one of the possibilities for the learner to end a challenge with the OLM, in this case by refusing to commit themselves in agreeing or disagreeing with the OLM's judgement.	WINDUP
AGREE	Used by the learners to notify their agreement with the OLM's judgement on the topic of discussion. AGREE is one of the possibilities for the learner to end successfully a challenge with the OLM. In this case, the OLM takes note of the acceptance of the claim by the learner (by triggering a WINDUP move) and sends an event to the Learner Model, signaling that the issue (i.e. the belief) has been agreed by the learner. This move in effect reinforces the judgement of the OLM on the belief.	WINDUP

(continued on next page)

Table 4.1: Description of the Dialogue Moves

(continued from previous page)

Move	Description	Follow-ups
DISAGREE	Used by the learners to state their disagreement with the judgement made by the OLM on the current belief. The disagreement is focused on one of the element of the argumentation pattern (claim, warrant or backing) and is dealt with by the OLM in the Challenge View (see section 4.2.3.1). Depending on the resolution of the disagreement, the OLM could either accept the challenge and trigger a WINDUP move, or remain unmoved by the learner's alternative statement and suggest a series of exercises to resolve the issue (trigger a SUGGEST move).	WINDUP SUGGEST
SUGGEST	Used by the OLM to suggest a series of exercises for the learner to perform in order to revolve a disagreement (in effect requesting more evidence before returning to the same issue later on). Based on the topic of discussion (the belief) and on the context of disagreement (claim, evidence, etc.), the OLM will send a request to the Tutorial component for generating an appropriate book. The suggestion for performing a new series of exercises before talking again about the issue is proposed to the learners for their acceptance or refusal.	AGREE DISAGREE
ACCEPT REFUSE	Used by the learner to accept or reject a suggestion made by the OLM. Two situations could arise where these moves are triggered: • in the UNRAVEL move, when the learner is suggested to explore one of the unresolved issue hold by the OLM. Acceptance of the suggestion will trigger a SHOWME move whereas rejection will trigger another UNRAVEL with the next unresolved issue. • in the SUGGEST move, when the learner is suggested to perform a series of exercises in order to resolve a disagreement. Acceptance will request that the Tutorial Component generate a new sequence of relevant exercises whereas rejection will do nothing. In both cases, the topic of discussion is marked as unresolved.	

(continued on next page)

Table 4.1: Description of the Dialogue Moves

(continued from previous page)

Move	Description	Follow-ups
WINDUP	Used by the OLM to formally end the challenge of its judgement by the learner. This move basically provides the learner with the summary and the outcome of the challenge that took place on the given topic. Depending on the chain of dialogue moves, the outcomes could be one of the following: • if the learners agreed with the OLM on its judgement (as stated by a AGREE move), the OLM now considers the issue as resolved. The relevant belief is requested for an update on the basis of reinforcement of the evidence. • if the learners disagreed with the OLM and had their challenge accepted by the OLM, then the issue is considered as resolved. The relevant belief is requested for an update on the basis of the new alternative statement made by the learners. • if the learners disagreed with the OLM but had their challenge refused by the OLM (by suggesting more exercises before looking at the issue again), the issue is considered as unresolved. • if the learners or the OLM decided to give up the discussion (respectively by MOVEON or LETMOVE), then the issue is considered as unresolved. The challenge part of the discussion is over and the navigation mode re-initialised; the learner could once again build a belief descriptor and explore further judgements.	LOST SHOWME BYE

Table 4.1: Description of the Dialogue Moves

4.4 OLM and Natural Language

Two issues have been considered for implementing any NLU/NLG in the OLM:

1. As mentioned in the requirement 6.6, widgets pointing and clicking on a GUI is a much easier, faster and less error-prone mechanism than linguistic referencing. This is not a surprise and is not specific to Open Learner Modelling.

2. Building a NLU engine for the OLM is a huge operation which requires, among many other things, to build a domain reasoner able to deal with the specific tasks and functions of the OLM. WP5 had a long experience in this topic and indeed provided such a reasoner for the content of LeActiveMath, namely differential calculus. But dealing with the OLM would have meant to run experiments, collect a corpus, build a reasoner, etc. for the specific usage of the OLM.

The gains for a NLG/NLU compliant interface had to be weighed against the time and resource required for an effective deployment of an interface. When we did it, it very quickly appeared that the cost for a fully dialogue-enhanced OLM was not balanced by the added value that such an interface might give us, especially in the narrow definition of the project. It is our belief that a dialogue-enhanced OLM is a full research project on its own.

Therefore we made the decision to significantly play down the natural language aspect of the OLM. First, the learner interactions with the OLM are handled solely through widgets at the GUI level, i.e. by pure direct manipulation. This meant that NLU was not required anymore from the OLM. Second, it was nevertheless considered important that the OLM should support some kind of natural language mapping to the elements displayed by and negotiated within it and that this "transcription" should be made as a complement of the main dialogue taking place in the GUI[8].

Three different aspects of the OLM are therefore considered in terms of natural language:

- the generation of an natural-language transcription of the learner's interaction with the OLM;
- the language used to refer to the content of the Learner Model (i.e. the belief itself, the various layers and their internal topics, the levels used to model the degree of ability of the learners on a given layer, etc.);
- the internationalisation of the interface (widgets and content).

4.4.1 Transcription of the dialogue moves

Every single dialogue move used in the OLM is associated with one or many NL-templates that are used to generate a natural language transcription of the interaction between the OLM and the learner. Each of these templates have been designed in order to precisely describe the current context of the learner's interaction; this is why a single dialogue move may have several templates to discreminate between various similar situation. The listings 4.1, 4.2 and 4.3 gives an extract of the templates used for the dialogue moves.

For example, the `PERHAPS` dialogue move is associated with three different templates: one to be used is the Learner Model does not held any belief on the given descriptor (`I'm afraid I know nothing about you on {1}`), one for delivering a value judgement about the belief (`I think your are {0} about {1}`) and one for delivering a score about the belief (`I think your are scoring {0,number,#.###} about {1}`).

The placeholders in the templates (materialised by the {n} token, where n represents the nth value to be introduced in the template) will be replaced by the proper values specified by the code using the template. In our example, the placeholder {0} will be replaced by the value of the claim (i.e. summary belief) made by the OLM, wether as a number or as a NL value judgement (see below). The placeholder {1} will be replaced by the reference to the belief descriptor.

4.4.2 References to the belief

The transcription of the dialogue move (and part of the main interface as well) needs to explicitly refer to some of the content of the Learner Model, elements like the belief itself (its descriptor), the various levels associated with each of the layers in the model, etc.

In this document, the beliefs have been referred to by their belief descriptor, represented by the following n-uple:

```
[domain, capes, competency, motivation, affect, metacognition]
```

[8] One of the reasons for this decision is that, despite the well-known adage, a diagram is not always worth one thousands words.

Listing 4.1: Extract of the NL templates used for the dialogue moves

```
# NLG templates for the dialogue moves
DlgMove.Agree.Accept      = Yes, why not.
DlgMove.Agree.Agree       = I agree with {0}.
DlgMove.Agree.Disagree    = I disagree with {0}.
DlgMove.Agree.Reject      = I don''t want to do that
DlgMove.Baffled.Help      = What do you mean?
DlgMove.Baffled.What      = I don''t know what to do now.
DlgMove.Baffled.Why       = I don''t understand why you think {0}.
DlgMove.HereIs.Evidence   = Here is the evidence for {0}
DlgMove.Perhaps.Ignore    = I''m afraid I know nothing about you on {0}.
DlgMove.Perhaps.Judgment  = I think your are {0} about {1}.
DlgMove.Perhaps.Score     = I think your are scoring {0,number,#.###} about {1}.
DlgMove.Quit.Bye          = Well, see you very soon ...
DlgMove.Quit.Prompt       = It was a nice talk but I need to go now.
DlgMove.ShowMe.Think      = Show me what you think I know about {0}.
DlgMove.Startup.LeAM      = Welcome my dear {0}. Why don''t you have a look at {1}?
DlgMove.Startup.OLM       = Welcome my dear {0}.
DlgMove.Startup.User      = Welcome my dear {0}. What brings you here today?
DlgMove.Unravel.NoIdea    = I can only advise you to explore the topics you are more familair with.
DlgMove.Unravel.Suggest   = Perhaps you should explore {1}.
DlgMove.Unravel.Urge      = We didn''t finish our discussion on {1} last time. Why don''t we have a look again?
DlgMove.WindUp.Accept     = This is how I think you should have judged me.
DlgMove.WindUp.Challenge  = In this case, what is your own judgement about the situation?
DlgMove.WindUp.Moveon     = Let''s change the topic of discussion, please.
DlgMove.WindUp.Resolved   = That''s OK, I will make sure to take this fact into account.
DlgMove.WindUp.Unresolved = Fine, but we will have to come back to this issue later.
```

Listing 4.2: Extract of the NL templates used for the dialogue moves (in French)

```
# NLG patterns for the dialogue moves
DlgMove.Agree.Accept      = Oui, pourquoi pas
DlgMove.Agree.Agree       = Je suis d''accord avec {0}.
DlgMove.Agree.Disagree    = Je ne suis pas d''accord avec {0}.
DlgMove.Agree.Reject      = Je ne veux pas faire cela.
DlgMove.Baffled.Help      = Que veux-tu dire?
DlgMove.Baffled.What      = Je ne sais pas quoi faire maintenant.
DlgMove.Baffled.Why       = Je ne comprend pas pourquoi tu estime {0}.
DlgMove.HereIs.Evidence   = Voila les preuves que je possède a propos de {0}
DlgMove.Perhaps.Ignore    = Je suis désolé, je ne sais rien à propos de {0}.
DlgMove.Perhaps.Judgment  = Je pense que tu es {0} à propos de {1}.
DlgMove.Perhaps.Score     = J''estime que ta performance sur {1} est de {0,number,#.###}.
DlgMove.Quit.Bye          = Bien, à la revoyure...
DlgMove.Quit.Prompt       = Sympa la discussion mais je dois y aller maintenant.
DlgMove.ShowMe.Think      = Montres-moi ce que tu crois savoir de moi à propos de {0}.
DlgMove.Startup.LeAM      = Bienvenu {0}. Pourquoi ne jetterais-tu pas un oeil sur {1}?
DlgMove.Startup.OLM       = Bienvenu {0}.
DlgMove.Startup.User      = Bienvenue {0}. Qu''est-ce qui t''amène ici aujourd''hui?
DlgMove.Unravel.NoIdea    = Pourquoi ne pas explorer les sujets avec lesquels tu es familier?
DlgMove.Unravel.Suggest   = Peut-être que tu pourrais explorer {1}.
DlgMove.Unravel.Urge      = Nous n''avions pas terminé notre discussion à propos de {1}. Pourquoi ne pas continuer maintenant?
DlgMove.WindUp.Accept     = C''est comme cela que tu aurais du juger mes performances.
DlgMove.WindUp.Challenge  = Dans ce cas, quel est le jugement que tu portes sur cette situation?
DlgMove.WindUp.Moveon     = Peut-on changer de sujet de discussion?
DlgMove.WindUp.Resolved   = Très bien, je vais prendre en compte ta décision.
DlgMove.WindUp.Unresolved = Bien, mais je pense qu''il faudra revenir sur ce point plus tard.
```

Listing 4.3: Extract of the NL templates used for the dialogue moves (in German)

```
# NLG patterns for the dialogue moves
DlgMove.Agree.Accept      = Ja, warum nicht.
DlgMove.Agree.Agree       = Ich stimme {0} zu.
DlgMove.Agree.Disagree    = Ich stimme {0} nicht zu.
DlgMove.Agree.Reject      = Das möchte ich nicht tun.
DlgMove.Baffled.Help      = Das verstehe ich nicht.
DlgMove.Baffled.What      = Ich weiß nicht, was ich jetzt tun soll.
DlgMove.Baffled.Why       = Keine Ahnung, wie es zu {0} kommt.
DlgMove.HereIs.Evidence   = Hier sind die Belege für {0}.
DlgMove.Perhaps.Ignore    = Leider keine Aussage möglich zu {0}.
DlgMove.Perhaps.Judgment  = Ihre Einstufung für {1}: {0}
DlgMove.Perhaps.Score     = Ihre Bewertung für {1}: {0,number,#.###}
DlgMove.ShowMe.Think      = Was weiß ich über {0}?
DlgMove.Startup.LeAM      = Hallo {0}, schauen Sie sich doch mal {1} an.
DlgMove.Startup.OLM       = Hallo {0}.
DlgMove.Startup.User      = Hallo {0}, wie geht's?
DlgMove.Unravel.NoIdea    = Vielleicht schauen Sie einmal nach Themen, mit denen Sie sich besser auskennen.
DlgMove.Unravel.Suggest   = Schauen Sie sich doch mal {1} an.
DlgMove.Unravel.Urge      = Die Diskussion über {1} wurde das letzte Mal nicht abschlossen. Vielleicht wollen Sie dort weitermachen?
DlgMove.WindUp.Resolved   = Okay, das wird in die Berechnung einfließen.
DlgMove.WindUp.Unresolved = Okay, das muß aber später noch einmal aufgegriffen werden.
```

Listing 4.4: Extract of the NL templates used for the belief layers

```
OLMTopicConfig.COMPET.Name    = Competency
OLMTopicConfig.COMPET.Level1  = Level I
OLMTopicConfig.COMPET.Level2  = Level II
OLMTopicConfig.COMPET.Level3  = Level III
OLMTopicConfig.COMPET.Level4  = Level IV
```

So, a belief about mathematical thinking on the chain rule will be represented by `[chain_rule,,think,,]`, where `chain_rule` refers to the identifier of the chain rule topic in the domain map and `think` refers to the identifier of the mathematical thinking topic in the competency map.

In the OLM, we are using a complete NL transcription for the belief descriptor, as it is the case in the transcription of the dialogue moves. It has to be noted that this is not always possible, in particular in some external representations like the Claim View. A complete NL description could be quite long and will not fit properly in the space-hungry view. In this case, we keep using the n-uple representation, with the obvious problem for the learner to map identifiers with topics. This issue has still to be resolved.

References to the topics of the various maps used in the Learner Model are also available with a full NL description that is used in the transcription of the descriptor. However, this is limited by the restriction imposed on the design of the topics maps, as explained in section 5.1.

Finally, the language used by the Learner Model (and the OLM) to diagnose the abilities of the learners are the levels (competency level, affective level, motivational level, etc.). In this document, there are referred as `Level I`or `Level IV`.

The question of how do talk about these levels to the learner is still an open question. The initial implementation of the OLM was basically to use this level terminology. It is a very convenient approach for an implementation point of view but not particularly useful.

The current implementation defines a very simple mapping between every level for each layer and a NL term used to represent it. For example, see figure 4.4, the terminology used for the competency keeps the "level" lexicon (since it originated from it), the terminology used for the affect used . The problem for this approach is that it assumes that all topics in a given map will be referred to by the same language. If that sounds not so bad from the competency point of view, it has proved to be particularly difficult with affective factors for example, where finding a common lexicon for factors so different like "pride" and "satisfaction" is not an easy task.

Therefore we are currently investigating the possibility for associating the lexicon used for a particular map, or even particular topics of the map, with the topic maps themselves.

4.4.3 Widgets and internationalisation

Internationalisation of the interface has been done very easily, using the features already provided by Java. All the strings used in the interface, whether for the widgets or the NL-templates used in the dialogue pane, are stored in separate files, in an attribute/value format. Internationalisation is therefore done by providing an extra file containing the translation of the value for every attribute. For example, the figure 4.1 shows some of the dialogue move NL-templates in english, whereas figure 4.2 shows the same templates but in French.

One important feature of the internationalisation framework is that placeholders are numbered (e.g. {0} for the first argument to be added, {1} for the second, etc.), meaning that the location of the placeholders in the template can be reorganised in order to accommodate syntactic and grammatical restriction of the target language (for example, the order of the arguments in the `DlgMove.Perhaps.Score` template has been changed to comply with a different structure for the

sentence in French). Another interesting is the possibility for defining the format of the given argument, especially numbers (see the {0,number,#.###} in DlgMove.Perhaps.Score template).

But one of the limitation is that the arguments are neither typed nor explicitly referred to. It means that the translation into a different language may turn out to be tricky if the translator has no idea of what is the expected argument in a given placeholder. Just giving the attribute/value list to translate is not enough; a strict definition of the templates and their arguments has to be given as well.

4.4.4 An Example of Dialogue

The table 4.2 shows an example of the Natural Language Generated output – in English – of a real interaction (i.e. discussion) between a learner (called **toto**) and the OLM.

Each piece of dialogue has been directly extracted from the OLM text pane. Information about the context of the interaction (i.e. which view is present on the GUI, which action the learner did, etc.) has been added in order to re-contextualise the dialogue. On the right-hand side of the output, the dialogue move responsible for generating the corresponding piece of dialogue has also been added.

A couple of points can be made on the basis of this output:

1. As a whole, the transcription of the dialogue works well, the articulation between the various dialogue moves going smoothly [9]. By keeping the arguments of the templates to a minimum and by using "abstract" references, it is possible to write a dedicated template in every language, without being too restricted by its structure and arguments (e.g. order of the arguments could be change from one language to another one).

2. Finding a NL transcription for the references to the OLM sources of information (i.e. belief descriptor, levels, Toulmin's elements) is still an issue that has to be improved. As can be seen in the dialogue, belief descriptors have been kept in their n-uplet format: the various attempts to give them an expanded NL form, so farm, have produced either sentences that were too long or un-informative ones. Referencing Toulmin Argumentation Pattern elements is half-sorted out: replacing the "claims" with the judgement it represents was easy and efficient but no such mapping has been found for the "warrants" yet (in the dialogue, WARRANT_7 refers to the 7th piece of evidence held by the system).

3. Since exploring a judgement made by the OLM consists of a repetitive sequence of the same actions (i.e. the BAFFLED-HEREIS pair of dialogue moves), it comes as no surprise that the NL output of the dialogue is also repetitive in places. Further user-testing will be needed in order to check if this issue is detrimental to an efficient use of the OLM; in such a case, alternative templates will be used, as well as extending the NL-generation mechanism for introducing variability in the output.

4.5 Design Issues

The main GUI of the OLM has been implemented with the Java SWING library[10].

It has to be noted that attention has been constantly paid to limitations introduced by the equipment (both hardware and software) potentially in use in real-life usage of LeActiveMath.

[9] At least in English and French; further user-testing in Spanish and German will be needed to make the same claim.

[10] Java SWING, see http://java.sun.com/docs/books/tutorial/uiswing/index.html

The first obvious restriction is that LeActiveMath is a web-based environment and the OLM is an applet embedded in a browser window. It means that a multiple-windows GUI has been ruled out for the implementation of the OLM in order to avoid confusion and overload. This decision has significantly impeded the potential of the GUI, in particular by preventing the use of simultaneous external representations and their supporting/complementing roles.

A second self-imposed restriction concerns the optimum size of the main GUI. The GUI has been designed knowing that, being an applet embedded in a browser, its size will be changed at any time, at the learner's initiative. We therefore made sure to use all the flexibility that `Java SWING` automatic layout mechanisms do support (i.e. widgets resized proportionally to their container's size, location of embedded panes automatically updated, etc.). But such automatic layout frameworks always have limitations, in particular when the GUI's window is reduced rather than increased. As it is not always possible to ensure a proper layout of a GUI in every possible configuration, we therefore introduced an optimum size for the OLM, i.e. an optimum above which we can guarantee that the layout of the various GUI's widgets will be adequately displayed but below which we cannot ensure it. This optimum size has been defined at 800x600[11], i.e. the minimal screen resolution commonly available on home computers.

The current layout of the various widgets and external representations in the GUI reflects that constant trade-off between readability, usability and informativity.

This is why, for example, we opted for a flip-card (i.e. multi-tabs) approach for the interface, why the NLG output pane is always visible at the interface (despite the obvious counter-argument that it does cost quite a lot of space for no necessary added value), etc. But it also means that space-consuming external representations have been difficult to implement and did need (and are still needing) fine-tunings for an optimal usage in a quite confined area. This is in particular the case for the graph-based external representation used in both for the alternative navigation interface (see section 5.3) and the Argumentation view (see section 4.2.1.2).

[11] At least this was the initial intention, until we realised that the implementation of the 800x600 limitations did not take into account the space needed by the browser's borders and by the operating system such as Windows' Task Bar.

DISPLAY: Screen Left: domain topics; Screen Right: Belief Descriptor (empty) *BUTTON CHOICES: I'm lost, I'm off*		
OLM	Welcome my dear **toto**. What brings you here today?	STARTUP
USER ACTION: I'm lost		
toto	I don't know what to do now.	LOST
DISPLAY: Screen Left: domain topics; Screen Right: Belief Descriptor [derivative,,think,,,] *BUTTON CHOICES: I agree, I disagree*		
OLM	Perhaps you should explore `[derivative,,think,,,]`.	UNRAVEL
USER ACTION: I agree		
toto	Yes, why not.	ACCEPT
toto	Show me what you think I know about `[derivative,,think,,,]`.	SHOWME
DISPLAY: Screen Left: argument view (claim node selected); Screen Right: claim view (level II) *BUTTON CHOICES: I'm lost, I'm off, I'm baffled*		
OLM	I think your are `Level II` about `[derivative,,think,,,]`.	PERHAPS
USER ACTION: I'm baffled		
toto	I don't understand why you think `Level II`.	BAFFLED
DISPLAY: Screen Left: argument view (data node expanded); Screen Right: claim view (level II) *BUTTON CHOICES: I'm lost, I'm baffled, I agree, I disagree*		
OLM	Here is the evidence for `Level II`	HEREIS
USER ACTION: select data node in argument view *DISPLAY: Screen Left: argument view (data node selected); Screen Right: Data view (pignistic ER)* *USER ACTION: I'm baffled*		
toto	I don't understand why you think `Level II`.	BAFFLED
DISPLAY: Screen Left: argument view (warrant nodes expanded); Screen Right: claim view (level II) *BUTTON CHOICES: I'm lost, I'm baffled, I agree, I disagree*		
OLM	Here is the evidence for `Level II`	HEREIS
USER ACTION: select "warrant 7" node in argument view *DISPLAY: Screen Left: argument view (data node selected); Screen Right: Warrant view (distribution ER)* *USER ACTION: I'm baffled*		
toto	I don't understand why you think `WARRANT_7`.	BAFFLED
DISPLAY: Screen Left: argument view (warrant nodes expanded); Screen Right: Warrant view (distribution and attributes)		

(continued on next page)

Table 4.2: Annotated NL-generation of a dialogue between the learner and the OLM.

(continued from previous page)

BUTTON CHOICES: I'm lost, I'm baffled, I agree, I disagree		
OLM	Here is the evidence for `WARRANT_7`	HEREIS
USER ACTION: I agree		
toto	I agree with `WARRANT_7`.	AGREE
DISPLAY: Screen Left: domain topics; Screen Right: Belief Descriptor `[derivative,,think,,,]`		
BUTTON CHOICES: I'm lost, Show me, I'm off		
OLM	That's OK, I will make sure to take this fact into account.	WINDUP
USER ACTION: build a new descriptor		
USER ACTION: Show me		
toto	Show me what you think I know about `[derivative,,solve,,,]`.	SHOWME
DISPLAY: Screen Left: argument view (claim node selected); Screen Right: claim view (level III)		
BUTTON CHOICES: I'm lost, I'm off, I'm baffled		
OLM	I think your are `Level III` about `[derivative,,solve,,,]`.	PERHAPS
USER ACTION: I'm baffled		
toto	I don't understand why you think `Level III`.	BAFFLED
DISPLAY: Screen Left: argument view (warrant nodes expanded); Screen Right: Warrant view (distribution and attributes)		
BUTTON CHOICES: I'm lost, I'm baffled, I agree, I disagree		
OLM	Here is the evidence for `Level III`	HEREIS
USER ACTION: I'm disagree		
toto	I disagree with `Level III`.	DISAGREE
DISPLAY: Challenge View		
BUTTON CHOICES: I agree, Let's move on		
OLM	In this case, what is your own judgement about the situation?.	WINDUP
USER ACTION: Let's move on		
toto	Let's change the topic of discussion, please.	MOVEON
DISPLAY: Screen Left: domain topics; Screen Right: Belief Descriptor (`[derivative,,communicate,,,]`)		
BUTTON CHOICES: I'm lost, Show me, I'm off		
OLM	Fine, but we will have to come back to this issue later.	WINDUP
USER ACTION: I'm off		
toto	It was a nice talk but I need to go now.	BYE
OLM	Well, see you very soon ...	

Table 4.2: Annotated NL-generation of a dialogue between the learner and the OLM.

5 Outstanding Issues and Future Work

5.1 The Domain Map

As mentioned in the specification of the Extended Learner Model (see D10 [17]), one of the outstanding issue was the question of the concept map used to represent the domain layer of the Learner Model. Several questions were at stake on that issue: the formalism to use for representing the knowledge, the location and availability of the map, its relation with the contents authored for LeActiveMath.

This issue proved to be a difficult matter to resolve; the solution currently implemented is a compromise that is not satisfactory for any party but which did permit the implementation of the various components, at least for the lifespan of the project. Basically, it consists in keeping the domain map separated from the content, implemented in the xLM and whose link to the content will be maintained top-down (i.e. from the domain concepts to the contents) – and by hand. The assumption made for the compromise was that, since this map was for the sole usage of the xLM, then there was no reason for interfering with the content per se, that both knowledge representation mechanisms could be independent.

Apart from the obvious design limitations of this approach, the compromise also turned to be unworkable from an integration point of view. The concept maps ARE NOT for the sole usage of the xLM and should be available – and used – in an integrated way, by all components of LeActiveMath.

Indeed, a huge gap does occur when the learners are practicing a particular exercise in LeActiveMath and want to explore the relevant part of the OLM: how could they know – or even guess – which concepts are involved behind this exercise? In short, LeActiveMath and the OLM don't speak the same language!

When the learners perform exercises in LeActiveMath, read some definition, explore some example, etc. they are talking with the language of the content (as materialised with OMDOC objects). When they access the OLM, they have to talk the language of the domain (as materialised by the topics in the domain map). At first inspection, this situation is not necessarily a problem, in particular since the OLM does provides a bridge between the two language when the evidence are introduced (i.e. the concept X is supported by evidence coming from contents Y and Z). But this is a one-way bridge only: in LeActiveMath, nothing in the front-end is preparing the learners for the tasks and the language that are awaiting them in the OLM.

5.2 Dynamic versus Static OLM

The assumption that is currently driving the implementation of the OLM is that the focus of the learners will be directed toward the actual state of the beliefs rather than toward the trajectory of their abilities; justifications will be supported by providing the learner with access to every individual evidence. Most of the External Representations described in this document provide the learners with an overview of the current state of the belief and not of its evolution across time.

Various (inconclusive) attempts to give access to a dynamic Learner Model have been tried. For example, an initial implantation of the Claim view offered the learner to switch between a representation of the current summary belief hold by the Learner Model and its complete evolution across time, i.e. with the introduction of each subsequent evidence (see figure 5.1).

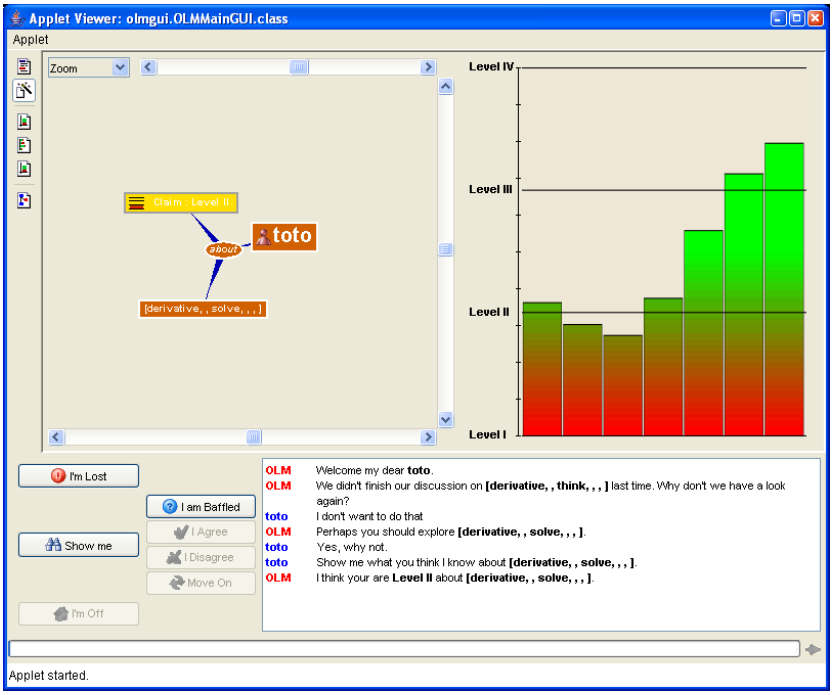


Figure 5.1: Displaying the evolution of the summary belief across evidence.

But by repeated usage and testing, it became quickly evident that this initial attempt for representing the dynamic process raised more issues than it did solve: the question of consistency across external representations (why restricting it the history to the summary belief only and not to others like the pignistic, the mass distribution, etc.); the question of controlling the dynamic representation (replaying the sequence forward and backward, stopping at the introduction of a given evidence, etc.); the question of supporting the translation between dynamic external representation (selecting a step of the sequence to access the related evidence).

5.3 Exploration and Navigation

As we have shown in the previous sections, exploring the content of the Learner Model (in other word, navigating through the OLM) is a complex but important task. The current approach, by explicitely building a belief descriptor from a list of available topics (see section 4.2.1.1), has been developed for its simplicity and straightforwardness.

Two issues have been considered. On one hand, such an interface should allow learners to find and select a belief descriptor (i.e. the underlying identifier behind any information in the Learner Model) that is relevant to their current goal, task and/or desire. On the other hand, the interface has to clearly and easily relate this descriptor to similar topics in the Learner Model (i.e. by highlighting the internal organisation of the topic maps and the relations between topics). If the current approach does support the first issue fairly well (by explicit listing all the possible topics, organised among the five distinct layers of the Learner Model), an alphabetical list of topics does not provide enough information for contextualising the belief.

Shortcuts and ad-hoc scaffolds on the existing interface (such as tooltips, access to description and definition of the terms used, etc.) could be implemented; alternative approaches are also to be considered. A first attempt for an alternative interface has been investigated, using dynamic

graphs to display and present to learners, not only all the relevant topics but also their connections and inter-dependencies (see figure 5.2).

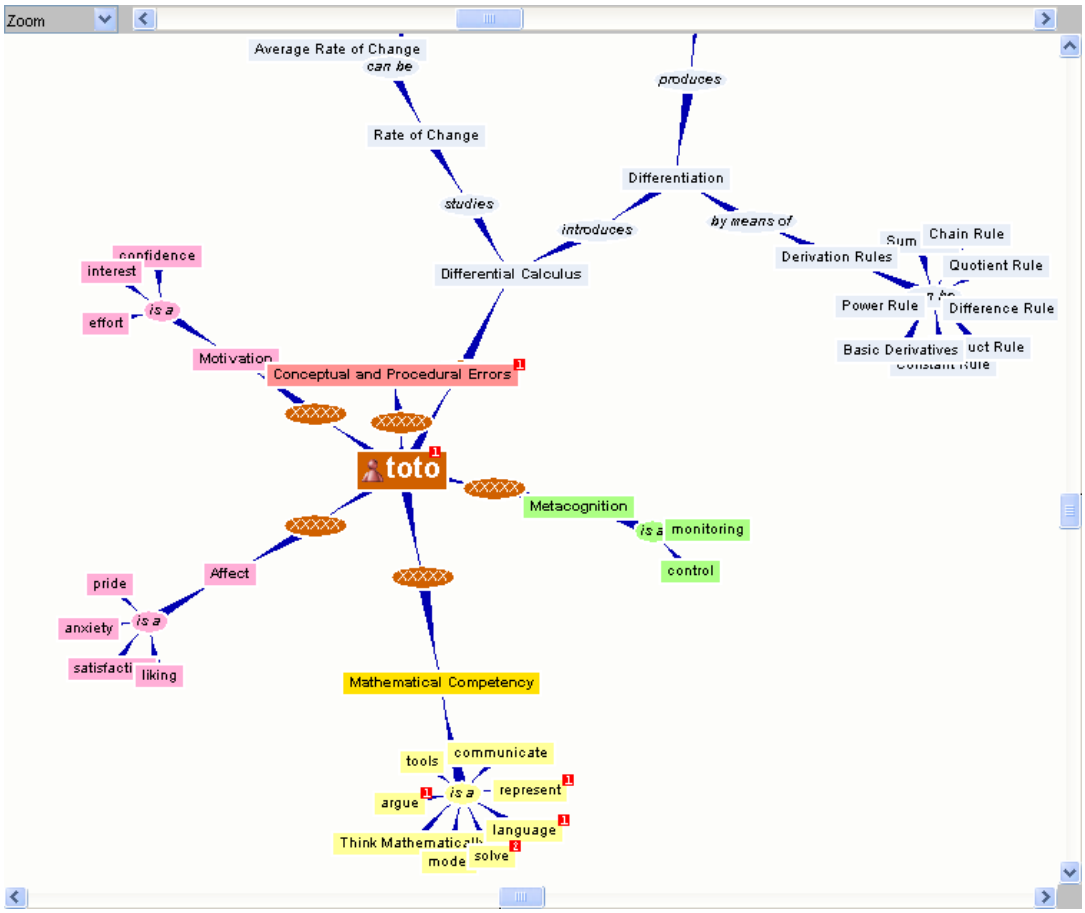


Figure 5.2: An alternative view for the exploration of the content of the Learner Model

The figure 5.2 represents the learner as the central node, with all the topic maps used by the OLM spreading from it: metacognition, affect and motivation maps are fully expanded; mathematical competencies are partially expanded, only the sub-competencies being hidden; the domain map (with its top-node *"differential calculus"*) is partially visible on the top of the figure.

Using such a "interactive map" will certainly increase the usability of the interface (introducing both topics and associations to develop the narrative of the map, dynamically hide/expande/collapse sections of the graph that are not of an immediate usage by the learner, etc.) But issues such as the potential unfamiliarity of a graph approach, the difficulty of manipulation, the whole readability of the approach, etc. have also to be taken into account to balance out judgement.

The graph, implemented with an open-source library[1], supports very useful features such as automatic layout (spring model), expansion/collapsing of nodes and sub-graphs on-the-fly, totally configurable and re-implementable UI (both representation and manipulation of graphs). It is already available in the OLM (by using the `View Topic Map` button on the main GUI) but has limited functionalities (only belief descriptors are dynamically added on-the-fly, no filtering of graph, expansion and collapse of nodes/subgraphes at learner's request).

[1] TouchGraph, see http://touchgraph.sourceforge.net/

5.4 Hiding and Restricting information

The issue of privacy in learner modelling (i.e. hiding part of the information that the Learner Model is holding about the learner) has been mentioned as a difficult topic straight from the beginning of the project (see requirement 5.8 in the introduction) and is a research question that has been emerging rapidly in the user-model/learner-model community.

The complexity and novelty of the design of the xLM (both the LM and the OLM) put us in a situation where the insights from similar works had a very small impact on our own work. Our design has necessarily led us to select information that we believe should be showable in principle. In a sense, the kinds of information selected are a superset of the kinds shown in other systems. The decision for the implementation of the OLM has been to present everything to the learner and then make use of user-testing and evaluation to consider some of the effects (or side-effects) of displaying such or such pieces of information and then decide on what restrictions to place on the available information.

Initial user-testing sessions allowed us to suggest some approaches for addressing this issue (i.e. what to hide and when). For example, despite the impact that displaying the meta-cognitive abilities of the learners will have, it is also clear that challenging them have no theoretical or empirical ground. On another level, some of the topics in the various maps used by the Learner Model are more of an internal use. This is the case for example of the sub-competencies, introduced for trying to discriminate between similar situation.

5.5 Complexity and Usefulness of the OLM

The final point to mention here is the whole question of the usefulness of the OLM.

As it should be evident after reading the relevant deliverables, the Learner Model implemented by WP4 is a quite complex component, building and updating a wide range of interconnected information about the learner's abilities. Therefore, it comes to no surprise that the OLM, if seen merely as the Graphical User Interface of the Learner Model, does reflect this complexity. Complexity that become even more evident once mechanisms for supporting learner's self-reflection, for explaining and challenging the Learner Model decisions are introduced in the picture.

On one hand, abridged representations of the internal data of the Learner Model can without a doubt be deployed at the interface, to insure a proper and efficient readability for the learner. This is for example the purpose of the Claim view, i.e. the externalisation of the quite straightforward and simplistic (but accurate) summary of a belief held by the Learner Model. On the other hand, by performing such an abridgment, a significant amount of information is (deliberately) not presented to the learner, therefore blurring the "logic" of the reasoning followed by the Learner Model when establishing its belief and, consequently, impeding the possibilities for the learner to efficiently challenge its decisions. A more detailed external representation of the internal data is therefore needed, as it is the case for example with the Certainty view of the belief. By presenting all probabilities for every range of levels, the ER clearly extends the potential for justifying the model but, as a consequence, now impedes its readability.

And finally, when the catch-22 situation above is allegedly resolved by the obvious solution (i.e. by introducing alternative external representations, whose complementary roles will support different purpose in different context), the sheer complexity of the mechanism required for organising the several alternatives, for deciding which representation learners should use to fulfil their goal, for modelling and managing such goals, expectations and fulfilment, seems to overwhelm any gain made.

The current state of the OLM, as described in this document, is a first sustained attempt in finding a trade-off between the two (apparently contradictory) aims of readability and justifiability.

The introduction of several complementing external representations (such as the summary belief, the pignistic function, the mass distribution, etc.), the use of a Toulmin-inspired argumentation framework for organising and controlling the usage of these representations, the supportive role that the natural-language "transcription" of the learners dialogue moves are all steps in reaching this objective.

Bibliography

[1] Vincent Aleven and Ken Koedinger. Limitations of student control: Do students know when they need help? In *ITS'00 - 5th International Conference on Intelligent Tutor Systems*, pages 292–303. Springer-Verlag, 2000.

[2] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[3] Susan Bull and Helen Pain. 'Did I say what I think I said, and do you agree with me?': Inspecting and questioning the student model. In Jim Greer, editor, *World Conference on Artificial Intelligence and Education*, pages 501–508. Charlottesville VA: AACE, 1995.

[4] Pierre Dillenbourg and John Self. PEOPLE POWER: A human-computer collaborative learning system. In Claude Frasson, gilles Gauthier, and Gordon McCalla, editors, *Intelligent Tutoring Systems: Second International Conference, ITS'92*, number 608 in Lecture Notes in Computer Science, pages 651–660, Montréal, Canada, 1992.

[5] Vania Dimitrova, John Self, and Paul Brna. The interactive maintenance of open learner models. In Susanne P. Lajoie and Martial Vivet, editors, *Artificial Intelligence in Education—Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration; proceedings of the 9th International Conference on Artificial Intelligence in Education*, number 50 in Frontiers in Artificial Intelligence and Applications, pages 405–412. IOS Press, 1999.

[6] Vania Dimitrova, John Self, and Paul Brna. STyLE-OLM—an interactive diagnosis tool in a terminology learning environment. In Rafael Morales, Helen Pain, Susan Bull, and Judy Kay, editors, *Proceedings of the Workshop on Open, Interactive, and Other Overt Approaches to Learner Modelling*, pages 25–34, Le Mans, France, July 1999. AI-ED'99.

[7] Benedict du Boulay, Rosemary Luckin, and Teresa del Soldato. The plausibility problem: Human teaching tactics in the 'hands' of a machine. In Susanne P. Lajoie and Martial Vivet, editors, *Artificial Intelligence in Education—Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration; proceedings of the 9th International Conference on Artificial Intelligence in Education*, number 50 in Frontiers in Artificial Intelligence and Applications, pages 225–232. IOS Press, 1999.

[8] Pentti Hietala and Timo Niemirepo. Collaboration with software agents: What if the learning companion agent makes errors? In B. du Boulay and R. Mizoguchi, editors, *Artificial Intelligence in Education: Knowledge and Media in Learning Systems. Proceedings of the AI-ED 97 World Conference on Artificial Intelligence in Education*, number 39 in Frontiers in Artificial Intelligence and Applications, pages 159–166, Kobe, Japan, 1997. IOS Press.

[9] LeActiveMath Partners. Exercise language. LeActiveMath Deliverable D7, The LeActiveMath Consortium, 2004.

[10] LeActiveMath Partners. Open architecture. LeActiveMath Deliverable D8, The LeActiveMath Consortium, 2004.

[11] LeActiveMath Partners. Requirement analysis. LeActiveMath Deliverable D5, The LeActiveMath Consortium, June 2004.

[12] LeActiveMath Partners. Structure and metadata model. LeActiveMath Deliverable D6, The LeActiveMath Consortium, 2004.

[13] LeActiveMath Partners. Design of guis. LeActiveMath Deliverable D23, The LeActiveMath Consortium, 2005.

[14] LeActiveMath Partners. Diagnostic functionalities. LeActiveMath Deliverable D30, The LeActiveMath Consortium, 2005.

[15] LeActiveMath Partners. Enhanced dictionary. LeActiveMath Deliverable D15, The LeActiveMath Consortium, 2005.

[16] LeActiveMath Partners. Formalized pedagogical strategies. LeActiveMath Deliverable D20, The LeActiveMath Consortium, 2005.

[17] LeActiveMath Partners. Student model specification (revised version). LeActiveMath Deliverable D10, The LeActiveMath Consortium, 2005.

[18] Antonija Mitrovic. Investigating students' self-assessment skills. In *UM'01 - 8th International Conference on User Modeling*, volume LNAI 2109, pages 247–250, Sonthofen (Germany), 2001. Springer-Verlag.

[19] J. Nielsen and T. K. Landauer. A mathematical model of the finding of usability problems. In *INTERCHI '93*, Amsterdam, The Netherlands, 1993. ACM Press.

[20] John Self. Bypassing the intractable problem of student modelling. In Claude Frasson and Gilles Gauthier, editors, *Intelligent tutoring systems: At the crossroad of artificial intelligence and education*. Ablex Publishing Corporation, Norwood, NJ, 1990.

[21] John A. Self. Dormorbile: A vehicle for metacognition. AAI/AI-ED Technical Report 98, Computing Department, Lancaster University, Lancaster, UK, 1994.

[22] John A. Self. Formal approaches to student modelling. In Jim E. Greer and Gordon I. McCalla, editors, *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, volume 125 of *NATO ASI Series F: Computer and Systems Sciences*, pages 295–352. Springer Verlag, 1994. Proceedings of the NATO Advanced Research Workshop held in Ste. Adele, Quebec, Canada, May 4–8, 1991.

[23] S. Toulmin. *The Uses of Arguments*. Cambridge University Press, 1959.

A User-Testing: A First Report

In December, user-testing sessions were run in Glasgow in order to gather information about the usability and usefulness of the OLM. They provided us with a couple of insights on the issues to improve – or even to change – in the OLM.

The number of testers is small, arguably enough to find the main problems [19]. Some of the problems identified by the testers have already been taken into account in the current version of the OLM; others, because of their difficulty or lack of obvious alternative, will be considered over the following months.

Below is the transcription of one of these sessions, where important remarks and comments made by the tester – or by the experimenters – has been highlighted. A summary of the improvements to consider is given at the end of the section.

OLM User Testing - Summary - 4 January 2006

Present: Iain Gibson (IG), Nicolas Van Labeke (NV) and anonymous tester 'Toto'.

NV sets the scene: Toto has done a few exercises in LeAM and now wants to explore the OLM.

NV: Your task is to understand what the OLM is, what you can do with it, and so on.

OLM is started up. NV explains more about what OLM basically does; Toto's task is to have a dialogue with OLM and explore what it believes. Several minutes pass as Toto tries to figure out front screen.

NV points out Bye and I'm Lost buttons. Toto doesn't notice anything else for a bit... [It's hard to figure out what to do without any guidance.]

Toto sees Domain menu and other menu headings. Goes into Motivation. NV points out hint boxes (tool tips) that are appearing.

Toto: What about Interest? *[Toto double-clicks on it. Interest box is highlighted. Show Me button also highlighted. Toto clicks Show Me. OLM doesn't understand.]*

NV: You have to ask about interest etc. with respect to some element of the domain.

Toto goes to Differential Calculus (domain) menu and chooses Derivative. Clicks Show Me. OLM understands but doesn't know anything.

Toto: Aren't there a load of sub-topics under Derivative? NV explains it is a valid choice since it appears in the Domain menu. It's just that OLM has no evidence on this question.

Toto chooses I'm Lost. OLM suggests Think, Derivative. The I Agree/Disagree buttons are highlighted. Toto presses I Agree. Screen changes to claim view.

IG: What does Toto understand by 'Think'?

NV explains since it isn't at all obvious. Think is one of the Competencies. (Toto didn't look at this menu earlier.)

Toto*: it says Competency on the box but Mathematical Competency on the menu *[NV also notices Domain/Differential Calculus discrepancy]

NV explains some of the theory behind competencies.

NV: sub-competencies shouldn't appear on the menu.

Toto: how did I get to Level II?

NV: you've done 4 exercises with a steady improvement over time.

IG: what do you think of the new information on the screen? The graph and the bar on the left?

Toto: the bar (showing Level) gives a positive sensation. It's good that my name appears quite large on the graph... what if I want to express my disagreement with something the OLM says? NV points out I'm Baffled option.

IG: two things to point out here. First, we couldn't initially see this 'about' node on the graph and Nikolas had to zoom in to see it. Second, what does Toto think is meant by Levels? Level I=good/bad?

Toto understands that there is a progression from Level I through to IV.

Toto: does Level II mean I have a problem, or that I'm less than halfway through the course?

NV: it's nothing to do with how far you are through the material.

Toto: it's not so encouraging then if it's reflecting my performance. So I haven't done a very good job so far?

IG: you might think that, but if you know the definition of the Levels and so on then that might comfort you to some extent.

Toto: would I have the chance now to do some more work and get up to Level IV?... does this mean I have a basic understanding, or does level II mean I have a deficiency of some kind? Does it need to be spelt out if this is satisfactory or unsatisfactory?

IG explains a bit about what Levels mean. Being at Level II might be a perfectly satisfactory state for someone quite new to the material. It's just that they haven't had the chance to do complex exercises testing whether they're at Level III or IV. This kind of thing isn't brought out here on this interface, but it should be explained somewhere within the system!

NV: so it doesn't mean you're 'crap' or 'good', maybe just that you haven't met higher level material.

Toto: can different users have different paths through the material and end up with the same Level bar graph?

IG: totally, LeAM is quite an 'exploratory' system. *[Explains a bit.]*

Toto: what if someone just does easy questions and gets them all right?

IG explains to reach Level IV you have to do challenging questions well. NV elaborates on the distinction between performance on individual questions and the competency level.

IG: we should move on...

Toto clicks on I'm Baffled. Clicks on Data. Clicks Tell Me More a few times to see different charts. Ends up on pignistic view. Thinks for a while.

NV: clearly you need more information, but can you guess what's going on?

Toto: there's some concentration on Level II. Does the Level IV row mean I've got no sign of Level IV ability at all, or just a small bit? *[There's a small bit of blue, but it could look like meaning zero.]*

IG: that a point: there's no scale on these charts.

NV explains a bit about the pignistic. The OLM thinks Toto is most likely to be Level II, then I, III, IV.

Toto: it's a bit suprising it could think I'm quite likely to be Level II but not have reached Level I?

NV and IG explain that it means maximum Level.

Toto: this might be quite obvious what it means to mathematicians etc. but a lot of people wouldn't find it at all clear.

NV: LeAM could be used at secondary, even primary school, so they'd have no chance of following it!

IG: we should look at the other screens and see how comprehensible they are...

We move onto the Certainty screen. (9 bars)

Toto: I'm finding this even harder... so there are combinations of numbers and different colours...

NV explains a bit. Green means certainty, red plausibility and white certainty of the negation.

Toto: I wouldn't have known there was any white. *[It's white on a white background!]* And again there's no scaling.

NV explains a bit about the appearance of ranges of Levels. Also points out why I, II, III, IV isn't there (100% certainty!).

Toto: but that'd act as a scale for the picture.

IG/NV acknowledge this is a good suggestion they hadn't thought of.

Discussion about exact numbers appearing as 'tool-tips'; good idea to keep them subdued as they're not so informative.

Point about need for explanation of these screens in final package documentation. IG still finds them hard to understand!

We move onto the mass function screen. Toto thinks and guesses about what it might mean.

Toto: this could be quite encouraging because there is something on I, II, III, IV.

IG: we should note that this set wasn't on the previous screen. Back then it was tautologous information but here it is something different.

NV explains a bit about what this all means; distribution of evidence, betting analogy. The I, II, III, IV bar represents 'ignorance' on the part of the OLM. So here the OLM is quite ignorant about you and you've got a good chance to go on and demonstrate your abilities.

Toto: what about Help boxes or balloons?

NV: a very good suggestion.

IG: It's 4:20 so we should really move on. Nikolas can guide us through what needs to be covered.

Toto: what are all these appearances of Warrant?

NV: this word needs to be changed!

Toto clicks on various warrants and eventually I'm Baffled on a Warrant. Notices performance=0.5 for the exercise and asks what this means. [This should be explained in documentation.]

Toto notices not all details for exercise fit in their boxes. It's important to be able to see it all for user's reflection.

Toto catches on that the warrants are the mass functions etc. for individual exercises and the Claim somehow combines them.

Discussion that graph could get cluttered and possibility of grouping items. We move onto I Disagree with the Claim.

Toto: Let's try saying I'm Level III. How confident am I... here it should say percentage or values between 0 and 1, not just 10 to 90 as it is. What about qualitative terms (very confident, quite confident etc.)

Toto thinks about 'intransigence' question. (Do you think I may still be right. Yes/no.) I'll have to say yes, won't I?

NV: not necessarily, e.g. I could say I'm Level III when I'm not totally confident that I'm exactly Level III but I know I'm more than Level I.

Toto: could it have a maybe option, or more than a yes/no choice at least?

Toto: Why does confidence have a continuous range and intransigence is boolean?

Toto: maybe the variety is quite fun though and keeps your attention!

NV: It could be adapted. It does seem a bit coarse just to have yes/no. ... I think textual choices (hiding the numerical values) for confidence would be better as well.

Toto: What if they don't want to answer this question [intransigence]?

NV: good point. *[At the moment you either confirm or reject the whole screen, but not questions individually.]*

Points for Further Discussion/Action

- Emphasise to partners the need for clear instructions on use of OLM and underlying concepts such as competencies
- Tidy up discrepancies between menu headings on LHS and box titles on RHS
- Remove sub-competencies from competency menu.
- Add numeric scales to bar charts.
- Make visible the 'invisible' white bars in certainty/plausibility chart.
- Consider help boxes in different parts of OLM.
- Change 'warrant' to a more intuitive word.
- Ensure full details of exercise can be viewed in the 'Show Item' view.
- Re-design confidence bar when user disputes a claim. Consider adding percentage signs or a move to qualitative terms like 'very confident'.
- The colour-gradient used in the Claim view for displaying the summary belief (i.e. spreading from red - `Level I`- to green - `Level IV`) is confusing, in particular when related to the colour scheme used in the Certainty view (i.e. green for certainty, red for plausibility): tends to support non-existent interpretation.
- Consider more than two choices for intransigence.
- Consider allowing user to refrain from answering individual questions on the dispute screen.
- The wording of the validation buttons in the Disagree view is not clear enough for the user to figure out what to do once a statement has been made.

B The Event Generator

As the implementation of the LM and the OLM made progress, the need for a tool able to generate and publish events through the LeActiveMath framework became more and more self-evident.

On one hand, when debugging or testing a particular procedure (such as the evaluation of an event), having to open a book, find a page containing a exercise relevant to this procedure, run the exercise by trying to figure out which is the correct answer, was too much time-consuming. On the other hand, the content currently authored for LeActiveMath does not contain exercise for every possible combination of parameters[1], nor does the exercise sub-system generate all the possible assessment of the learner's performance on the exercise[2]. These limitations are a problem when the implementation of the diagnosis procedures of the LM, which are supporting all possible combinations, need to be tested.

The Event Generator currently supports all the events that are intercepted and interpreted by the LM (events such as `ExerciseStartedEvent`, `ExerciseFinishedEvent`, `SelfReportEvent`, etc.). It main interface, see figure B.1, uses a property-value sheet in order to display the relevant parameters of the event and control the edition of their associated values.

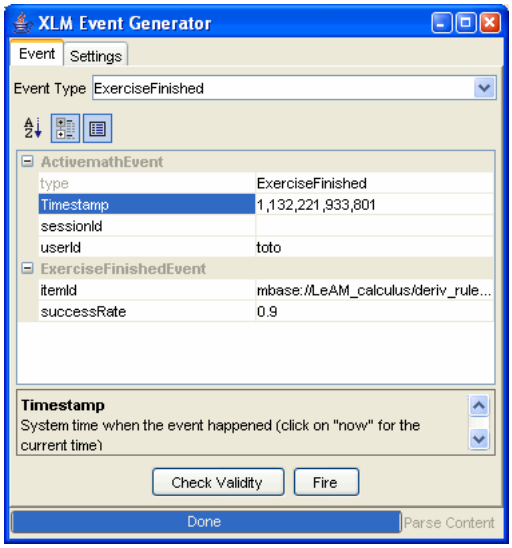


Figure B.1: A snapshot of the Event Generator

Simple type properties such as the user name (stored as a string), the success rate of an exercise (stored as a double) can be directly edited in the corresponding cell. More complex properties are supported by dedicated editors.

For example, the identifier of a `MBase` item, despite being a simple string, is in fact selected in a dedicated dialog box – see figure B.2(a) – which contains all the learning objects currently available in LeActiveMath, sortable not only by their `MBase` identifer but also by their type

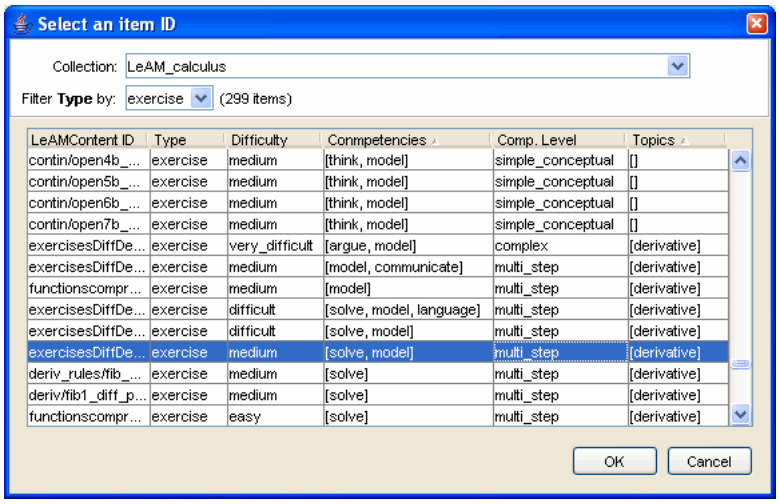
[1] For example, there is a strong correlation between the the difficulty of an exercise and its competency level, meaning that it is very difficult to find an exercise of difficulty `medium` and competency level `complex`; if a third property like competency is added to the constraints, then no exercise at all may exist.

[2] For example, in many exercise, especially the MCQs, the success rate generated by the exercise sub-system is either 0 (i.e. wrong answer) or 1 (right answer); despite being absolutely correct, it is nevertheless restricting when the behaviour of the LM need to be assesses in the case of a success rate of .5 or .75.

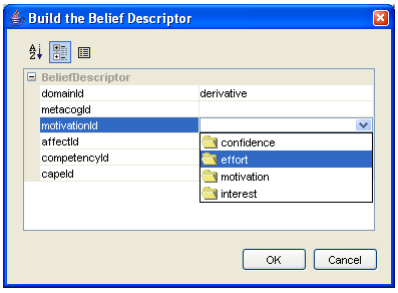
(exercise, theorem, etc.), their difficulty, the associated competencies and competency level and the related topic(s) from the domain map used by both the OLM and the LM. The interface gives us the possibility to select any particular exercise corresponding to any combination of competency-difficulty-topics etc..

Another example is the editor for a belief descriptor – see figure B.2(b) – as used by the `OLMMetacogEvent` and `OLMChallengeEvent` events. Each of the layer of the descriptor (i.e. domain, competency, metacognition, etc.) are specified by a popup list containing all the topics automatically extracted from the relevant map in LeActiveMath.

An extension of the tool is now considered in order to be able to generate, save and fire a complete script of events. Such a feature could be used in conjunction with "learner stereotypes" for testing the adaptability of LeActiveMath.



(a) LeActiveMath Content Selector



(b) Editor for Belief Descriptor

Figure B.2: Snapshots of various ad-hoc editors designed for the Event Generator