ALT-C 2008 Research Paper

Authors: Susanne Neumann
Petra Oberhuemer

Address for correspondence: University of Vienna
Centre for Teaching and Learning
Porzellangasse 33a
1090 Vienna
Austria
susanne.neumann-heyer@univie.ac.at
petra.oberhuemer@univie.ac.at

Bridging the divide in language and approach between pedagogy and programming: the case of IMS Learning Design

Abstract

Even though the IMS Learning Design (IMS LD) specification has offered a way for expressing multiple-learner scenarios, the language thus provided is far from the language teaching practitioners use. To bridge this divide, we have developed IMS LD authoring software that translates from the learning designer perspective to the technical perspective. To aid adequate software developments, an analysis was performed to identify uses of level B properties in expert units of learning. In a second analysis, which is described in this paper, these uses were matched with demands of typical pedagogical methods. Some restrictions of the IMS LD specification are pointed out in this regard. As an outcome of the analyses, interfaces employing pedagogical language were integrated in the IMS LD authoring software in order to provide teaching practitioners access to level B functionalities despite their highly technical nature.

Background

The language of IMS Learning Design

The IMS Learning Design (IMS LD) specification has presented a standardized language to describe multi-learner settings. It is the only specification that places the activities of participants rather than the learning content into its focus. This approach is meant to provide, among others, pedagogical flexibility, formalization and reusability of models of learning (Koper, Olivier & Anderson, 2003b).

To properly use the language of IMS LD, the learning designer follows a basic two-step process. First, the *components* of the learning design are specified such as the participating *roles*, learning and support (i.e. teaching) *activities*, and *environments*, which contain learning objects and learning services such as chat or forum. Second, these components are temporally orchestrated within the *method* to indicate what role performs what activity at what point in time. These components and method are integrated in an eXtensible Markup Language (XML) file, the imsmanifest, which holds the learning design. This XML-file is then packaged in a unit of learning along with any other content needed for the learning and teaching activities. The unit of learning may be imported into any learning management system that is capable of interpreting IMS LD such as dotLRN[1].

IMS LD is set up in three increasing levels of complexity to phase implementation efforts: levels A, B, and C. The simplest of these three levels, i.e. level A, is used for designing simple linear sequences of learning and support activities. The two-step process described above is employed to create units of learning at level A. Nevertheless, getting used to the terms and technical requirements is not trivial for learning designers, who are not necessarily accustomed to demands of technical setups.

Level B allows more flexible arrangements such as the individualization of learning paths and activities. The extended possibilities for designing learning sequences at level B, however, require an even extended level of technical knowledge, as level B concepts are essentially programming concepts.

[1] http://dotlrn.org/

To provide individualized learning, level B uses the additional component *properties*, which are data containers. Any type of data can be represented with properties, for instance, textual answers that were given to questions, files that are uploaded from the local file system, or numeric answers and assessments. These properties are being monitored by *conditions*, which are part of the method section of the imsmanifest and which regulate events within the unit of learning depending on how the values within the properties change. Typical applications for conditions are showing additional learning activities once the learner has uploaded a file, providing individualized feedback depending on the score that a learner achieved, and showing the learner comments of the tutor as soon as she has finished writing them.

Level C, finally, integrates system-generated *notifications* that may completely overrule other elements specific to levels A and B of the learning design. An example of a notification is that the tutor is informed once a learner has finished a certain activity.

The often reported problem with the IMS LD specification is that the so provided language is distant from the language teaching practitioners use in their daily routine (Beetham, 2004; Griffiths & Blat, 2005). Additionally, the representation of realistic learning models necessitates the use of level B concepts (Barrett-Baxendale, 2007), which in turn requires programming knowledge (Heyer, Oberhuemer, Zander & Prenner, 2007).

As practitioners cannot be expected to be proficient in programming concepts, the gap thus opened may be described as follows: On the one hand, IMS LD created the possibility to model activity-centered learning designs that also holds the promise of inspiring more learner-centered designs for e-learning and blended learning scenarios. On the other hand, a seemingly insurmountable barrier is placed before teaching practitioners, who are to apply the IMS LD concepts without having attended courses for basic programming. We aimed at closing this divide by providing software that integrates translations mechanisms from the learning designer perspective to the technical perspective of the specification. This way, teaching practitioners could design without being concerned about the technical details of creating IMS LD units of learning. The analyses that were performed to support adequate software developments are presented in this paper. Since level B is considered necessary for adequate representation of learning scenarios, the focus is here placed on the level B developments.

Previous work

Early wizard design

Taking a scaffolding point of view, the decision was made to create a wizard design that would be appropriate to guide learning designers when employing level B concepts. This wizard was meant to be implemented in a graphical IMS LD authoring tool – the Graphical Learning Modeller (GLM) (Neumann & Oberhuemer, 2008). The GLM visualizes sequences of learning and support activities in a graphical workspace and automatically detects and writes the needed elements for the IMS LD *method*.

Our initial hypothesis was that learning designers approach the construction of level B units of learning differently than the programming perspective requires: They would first think of the activities and what should happen to or within these activities. For instance, the learning designer would see the activity “Discuss results” in the workspace and would decide that this activity shall only be visible to the learners if the learners have submitted their solutions to a problem in a previous activity. In this way, the learning designer would first pay attention to the activity “Discuss results”, and then formulate what will happen to this activity according to the determining influences. The programming expert would approach this task from the opposite direction, first specifying the influences, and then formulating the result. This

difference in approaches between learning designers and programming experts takes place at the micro level (i.e. activity level). Sodhi, Miao, Brouns & Koper (2007) have similarly pointed out a difference of approach at the macro level (the unit of learning perspective), i.e. that IMS LD authoring software may support a top-down (starting with general ideas and theories on the learning approach) or a bottom-up approach (starting with the specification of granular components), arguing that learning designers prefer the top-down approach.

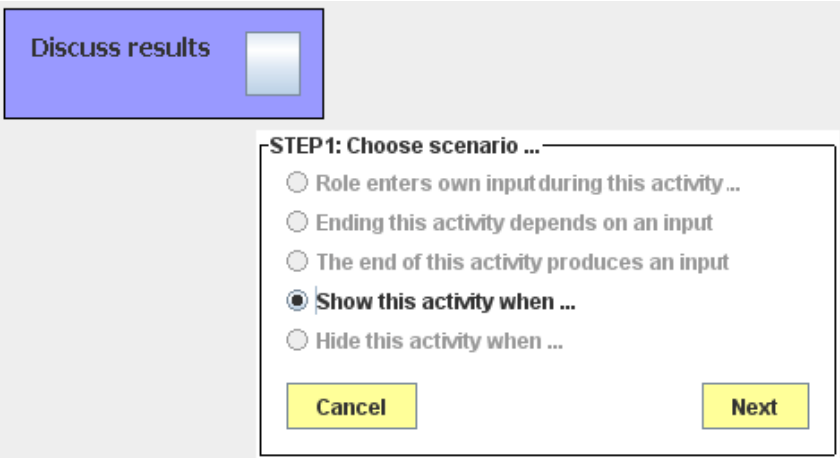

Figure 1: First design step within an early IMS LD level B wizard.

The wizard was meant to support the learning designer perspective. The result of taking this perspective is shown in Fig. 1, which depicts a screenshot of an early design of a level B wizard (implemented as a stand-alone Java animation). Directly at the activity in the workspace (represented by the box in the upper left corner named “Discuss results” in Fig. 1), the learning designer gets the opportunity to specify settings for this activity using the interactive button within the activity box (cp. Fig. 1). Upon pressing this button, the wizard expands and offers design choices for this particular activity. If the learning designer wishes that the activity “Discuss results” is only to be shown once the solutions have been submitted in a prior activity (not depicted in Fig. 1), then s/he selects “show this activity when…” from the wizard choices.

Three test users, who were all experts in technology-supported learning but non-experts in IMS LD, used this animation of the wizard to design three learning scenarios that were verbally described for them (they could read these from a piece of paper). The test users were asked to manipulate the wizard in such a way that the described learning scenario could be accurately arranged.

For the total nine test scenarios (three scenarios for three users), the test users successfully adjusted the wizard for eight of the scenarios. However, the test users reported that the wizard was awkward to use, and that it did not at all resemble the way they would go about setting up a learning situation. They stated that the wizard was oriented towards programmers, not educators.

Technical perspective on property uses

As the first wizard design proved an unsuccessful concept, a more thorough perspective on level B was needed. To improve upon the first wizard design, the goal of a second analysis was to recognize patterns within property uses in regard to their pedagogical purposes. In this second analysis, we looked at openly accessible units of learning that were designed by IMS

LD experts. These units of learning were taken from the IMS LD Best Practice and Implementation Guide (Koper, Olivier & Anderson, 2003a), the DSpace repository[2], and the Learning Design Handbook (Koper & Tattersall, 2005). Only units of learning that were at least level B were included in the analysis. Of each unit of learning, the imsmanifest and the accompanying resources (physical files like activity descriptions) were analyzed. Specifics of the analysis were described in Heyer et al. (2007); a summary is given here.

The analysis consisted of two parts. For the first part, the uses of properties were recorded from a technical point of view. For each property, the property-type, data-type, and technical place of use (e.g. condition, learning activity, resource of type imsld content[3]) were written in a spreadsheet application. Each use of the property was recorded separately because in this way, patterned steps in using the same property for similar pedagogical purposes became visible. For instance, a property's value was first changed within an activity description (a resource of type imsld content) using the global-element *set-property*[4]. This change in value was recognized within a condition, which was monitoring this property, and which in turn triggered another activity to be shown inside the unit of learning. Within this other activity, the entered value of the property was displayed. This represents a pattern of steps.

The second part of the analysis focused on the pedagogical side of property use, i.e. keeping track of the pedagogical purposes that properties were used for. Examples for pedagogical uses were the display of an opinion that a participant had written during runtime, earning points for completing a learning activity, or selecting one of several choices to identify one's own prerequisite knowledge level. The identified pedagogical uses from this analysis can be viewed in the first column of Table 1.

The two aspects of the analysis were then combined to identify how specific property-types and data-types were used regarding certain pedagogical purposes. The goal was to match the patterns of steps to specific pedagogical purposes. The analysis showed that the most common usage of properties was to have properties carry information that the participants of a unit of learning entered or changed during runtime (Heyer et al., 2007). This use clearly dominated all other purposes of property uses. Within this dominant use, a great portion of properties was of data-type *string* or *text*, indicating a strong focus on textual interactions.

As another outcome of the analysis, patterned steps for property uses were identified. This aided the wizard design because each set of steps could be packaged and matched to a particular pedagogical purpose. This way, each package with a pattern of steps is offered to the learning designer as a single entry point in the authoring software interface, reducing the number of interfaces needed.

Finally, we identified that the IMS LD specification neglects one of the goals for XML usage set out by the W3C consortium, namely that "XML documents should be human-legible and reasonably clear" (W3C, 2006). Properties are defined in the imsmanifest, but their values are often changed in files outside the manifest. To keep track of a property's use is a wearisome endeavor, considering that each property may be used in any one of the files accompanying the imsmanifest. The search for a property's use becomes something of a search for the needle in the haystack, lowering the ability to read and understand the XML.

---

[2] http://dspace.learningnetworks.org/handle/1820/16/browse-title

[3] IMS LD allows five types of resources: web content, imsld content, person, service facility, or dossier. For changing and viewing values of properties, imsld content is required (Koper et al., 2003b).

[4] With *set-property*, a specified property-value may be set by the user (Koper et al., 2003b).

Despite the promising results obtained from the analysis, the focus remained on the technical point of view. The need arose to complement this analysis with a pedagogical perspective in order to better perceive the potential of level B. In the following section, the main analysis for providing a pedagogical perspective on level B functions is described.

Analysis of level B from a pedagogical perspective: method & results

The goal remained to materialize the potential that level B holds, but to capture this potential more strongly in the language of teaching practitioners. As we had already identified the property relationships (what property-type and data-type go with what kind of pedagogical use), and the patterns of steps, we extended the analysis to see what had not been expressed in the expert units of learning.

The property uses identified in the previous analysis were placed into one dimension of a two-dimensional matrix (Table 1); the other dimension constituted a set of twenty generally known pedagogical methods derived from literature (compiled in Heyer, 2007). The setting for the methods (e.g. face-to-face, blended, online) was seldom explicitly stated; presumptively, face-to-face was the implicit setting for most methods. The granularity of the pedagogical methods varied from time and resource intensive methods like problem-based learning (Nelson, 1999) to short duration methods like think-pair-share (Harvard Project Zero). The number of pedagogical methods was reduced for the depiction in Table 1 due to a lack of space for adequately showing both dimensions.

For each pedagogical method, the granular activities that the participants would perform were identified. From these individual activities, it was concluded whether level B functionalities were needed to perform them or not. The analysis consisted of three parts: matching the previously identified property uses with demands of pedagogical methods, identifying new property uses from pedagogical methods, and identifying pedagogical needs that could not be properly expressed using IMS LD. The following sections explain the three parts in more detail.

Matching identified property uses with demands of pedagogical methods

For the first part of the analysis, the property uses from the previous analysis were placed in the first column, while the common pedagogical methods were placed in the top row. Take notice that not all the pedagogical methods included in the analysis are shown in Table 1, only a random set. Whenever an activity of a pedagogical method would require a property use from the list, a cross was placed in the mutual cell.

The first result from this part of the analysis confirmed a result from the previous analysis: text-based interactions were the most common property uses. Although merely showing a portion of the analysis table, Table 1 exemplarily demonstrates this finding as most crosses appear in rows of property uses relating to a textual interaction.

Table 1: Matching property uses with demands from pedagogical methods.

Pedagogical Methods ——————— Pedagogical Property Uses identified in expert units of learning	Problem-based learning	Project-based learning	Case-study method	Online Role-Play	Explore-Describe-Apply	Jigsaw-Method	Think-Pair-Share	Problem-Solving & Reflection	Brainstorming
Automatic check, whether an entry into an interactive field has taken place									
Automatic check whether given answers in a multiple-choice test are right									
Automatic display of feedback when a question was answered correctly									
Automatic display of feedback when a given answer to a question was wrong									
Automatic comparison of a given answer from free entry with a prespecified value									
Writing a comment	X			X			X		
Choosing an answer from a list or a multiple-choice question									
Answering a question by writing free text into a text field	X							X	
Giving signal (e.g. by pressing a button) that free text writing, or other free data entry is finished									
Providing different learning paths, depending on previous results or choices									
Individually choose preferences (e.g. knowledge level, preferred learning style etc.)									
Taking a vote (from prespecified answers)				X					
Giving personal written feedback to another person's free entry or answer				X	X				
Display of an (activity | resource | environment | class) depending on predefined conditions									X
Hiding of an (activity | resource | environment | class) depending on predefined conditions									
Manual (non-automated) checking of an answer from free text entry									
Display of an answer that was chosen from a list of choices or that was given via free text entry	X	X	X	X			X	X	X
Grouping of input fields (common display of joint fields)									
Giving a signal (e.g. by pressing a button) that an activity has been finished									
Earning points for finishing an activity									
Automatic calculation of the average of earned points									
The person sees a specific (activity | resource | environment | class) depending on the number of points earned									
Automatic calculation of the total points earned									X

A second finding was that many of the property uses previously identified from expert units of learning were specifically directed towards e-learning settings. Many of the previously identified property uses therefore received no crosses from pedagogical methods. Among

them were such uses as “earning points for finishing activities”, or “automatic check whether given answers in a multiple-choice test are right”. IMS LD is aimed at expressing any pedagogical approach (Koper et al., 2003b). However, it seems that the technical language has a tendency to target e-learning settings. One of the reasons for this may be that e-learning settings more readily justify the effort of creating IMS LD conformant units of learning than a face-to-face setting would. When expressing a face-to-face setting in IMS LD, the design is permitted to contain errors as there are no consequences for these errors, while the execution of an error-prone unit of learning in a learning management system would prove disastrous. For the same reason, face-to-face settings don’t need to be precise about using level B concepts as there are no consequences for employing or omitting them.

Identification of new property uses derived from pedagogical methods

The second part of the analysis was to identify new applications of properties that were not contained in the expert units of learning. The new property uses identified here are not necessarily employing new property types or data-types outside the ones used in expert units of learning. Rather, a new pedagogical name was attributed to property uses even if the technical setup of the property was identical. With these new names, multiple accesses to property uses may be provided. Examples of new names for property uses are:

- Authoring a (long) text
- Submitting a report (e.g. for results of group work)
- Keeping notes during reading, or during a discussion
- Writing a reflection
- Writing a justification/giving rationale
- Specifying a time limit for an activity while the unit of learning is running, e.g. for discussion time in a think-pair-share method or for idea collection during brainstorm
- Having the opportunity to ask a question (without request)
- Learners assign each other marks for work performed
- Separating identified problems into important and less important ones.

Identification of pedagogical uses that could not be expressed using IMS LD

Lastly, we kept track of requirements in the pedagogical methods that IMS LD was not capable of expressing. Problems with expressing pedagogical methods in IMS LD always arose when a change of plans would occur during runtime, i.e. when a new course of action that was previously unanticipated is taken. At this time, IMS LD provides no way of incorporating unanticipated courses of action into a running unit of learning as every element must be defined beforehand during design time. This may be the biggest letdown of IMS LD.

The interpretation of pedagogical methods into IMS LD is a grey zone, as the degree of expression is highly dependent on the imagination as well as IMS LD knowledge of the learning designer. S/he could either be stringent or relaxed when interpreting pedagogical methods for setup in IMS LD. For instance, one pedagogical method in our analysis specified that teams create activities, which are then assigned to and performed by other teams of the same unit of learning. In a stringent IMS LD interpretation, this setup may not be accurately expressed in IMS LD, since activities are by definition always specified beforehand during design time. In a looser interpretation, however, the learning designer could design the unit of learning in a way, where the newly created activity is written into a file, and the file is then uploaded during runtime into the unit of learning. The space for uploading the file must be defined during the design time.

Stating that unanticipated actions could not be expressed in IMS LD, we further provide an incomplete list of pedagogical methods that present difficulties when being expressed in IMS LD. The difficulties and some possible technical setups are described for each element.

- *During runtime, members of a group assign each other roles, which are also specified with a name and functional description by the team members.* The typical IMS LD interpretation of "role" cannot be used to implement this activity as all roles need to be defined during design time. In a workaround situation, initially empty properties would be designed that will carry the names of the roles once participants enter them. Using a *monitor service*[5], the name of the role could be displayed next to another property of data-type *boolean*[6]. For each person, the checkbox (boolean property) is marked in order to associate a person with a role. A condition in the method may then check what property was set for the person, and read the value of the property (i.e. the name of the role) accordingly. At any place, where the role is to be displayed, the global element *view-property*[7] has to be integrated in the resources of type imsld content beforehand so that the new value adjusted by the condition can be shown to the participant. This setup is hypothetical and has not been tested.
- *Results worked out during runtime (e.g. solutions to a problem) are to be classified into categories that are also defined during runtime.* Only a complicated setup could be used to design this activity according to IMS LD. During design time, the number of results to be submitted and the number of categories to be defined should be known for defining the corresponding properties. Most likely, the number is unknown beforehand, however. A random number of properties is thus specified so that each property may hold one result. Some of these properties may remain empty, but will be displayed nevertheless. The most difficult part is to assign a result to a category. The space herein does not suffice to offer the description of an idea to accomplish this complicated setup but may be similar to the description provided in the previous bullet.
- *Members of a team carry out activities that were assigned to them from a different team during runtime.* As explained before, this setup is not feasible in the stringent IMS LD interpretation but may be worked around using a file upload or text box, where the activity description is entered. This setup is feasible as long as the newly created activities do not integrate further activities that were to require additional properties.
- *Learners are to create a concept map with their ideas.* This can only be attained by creating the concept map in an external application and uploading this external file into the unit of learning. A property must be provided that will hold the file.
- *An activity or procedure has to be repeated as many times as there are groups present.* The problem is that at design time the number of participating groups is unknown. If the number of repetitions was known beforehand, the design could be set up correctly. However, if the number is unknown, then an arbitrary number of repetitions cannot possibly be designed (see also van Es & Koper, 2006).

---

[5] The monitor service provides a facility for users to look at their own properties or those of others (Koper et al., 2003b).

[6] Represents binary logic, e.g. on/off, true/false (Koper et al., 2003b).

[7] With *view-property*, a specified property-value may be viewed by the user (Koper et al., 2003b).

Implications for GLM design from analysis results

Applying the analysis results to software development, the idea of providing level B functionalities via a wizard was exchanged for the idea of offering an additional design area in the interface of the graphical IMS LD authoring software GLM. The special design area is called *Interactions* (cp. Fig. 2), and the learning designer receives a note when first clicking on this tab that informs about the types of design elements offered, namely for integrating design elements that allow interactive contributions of the unit of learning participants during runtime. The property uses that were captured in pedagogical language are provided as design choices, and thus build the entry points to level B functionalities.

Contribution

Fig. 2 shows a screenshot of the restructured level B interface, which was implemented in the GLM. To use interactions, the learning designer first picks the interaction s/he wishes to use (currently, four types are implemented: text work, uploading files, question & answer, and multiple-choice test), and creates a new instance of this interaction. In Fig. 2, the instance of the text work interaction is “Your best learning experience”. Each interaction is split into several steps, which may be individually dragged and dropped onto the activities in the workspace (pictured on the right in Fig. 2) to indicate where the interactions take place. For instance, the step “writing/editing the text” is dragged and dropped onto the activity, during which learners are to write about their best learning experience, in this case the activity “Reflect former learning experiences”. There, the learning designer has the opportunity to insert instructions and make adjustments to the type of text writing, e.g. whether each learner writes an own text, or whether all learners in that activity contribute to a common text.

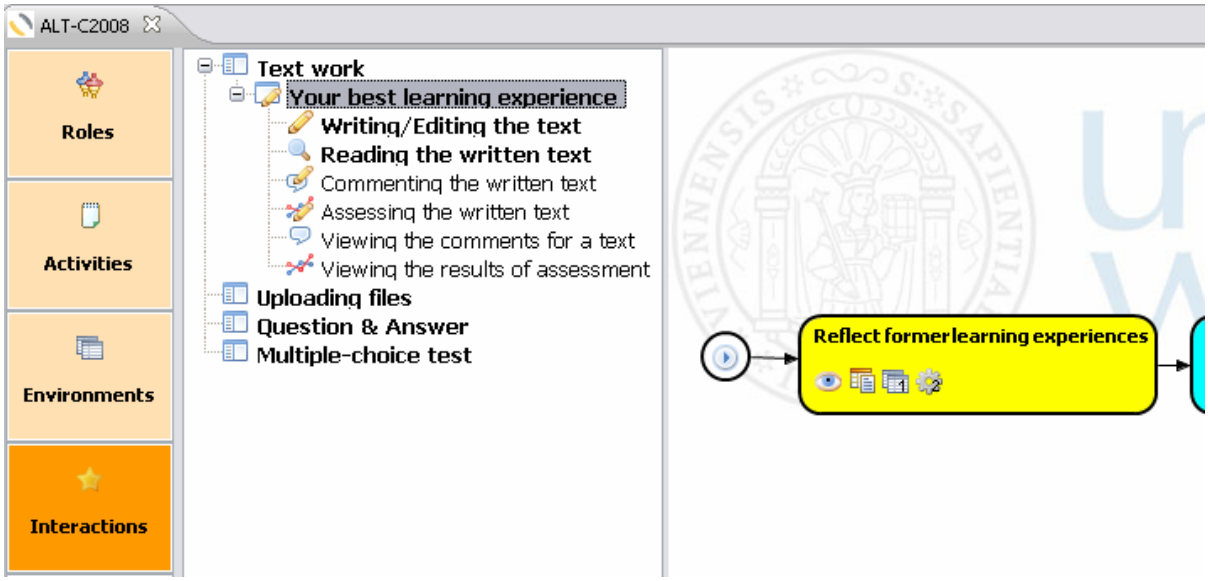

Figure 2: Partial screenshot of GLM software with implemented level B concepts.

All specified properties and conditions that are needed to execute the interactions are automatically written by the GLM software. It draws on algorithms that were derived from the analyses’ results presented herein. Since the analyses did not cover the entire potential that level B holds, the GLM offers a portion of level B functions. This set of functions is being continuously expanded.

Evaluation

Evaluations for the level B design are ongoing, while evaluations at level A have been almost completed. Evaluation results collected for a previous version of the GLM are shown in Fig. 3; more detailed information on the types and results of GLM evaluations can be found in Neumann & Oberhuemer (2008). Fig. 3 summarizes the feedback of four industrial test bed partners that use the GLM as part of their learning process configurations. They rated the GLM according to seven dialogue principles specified in the ISO 9241/110 standard (Faltin, 2008). The GLM was assigned above neutral ratings (above 3) for suitability for the task, controllability, conformity with user expectations, and suitability for learning. If the values for self descriptiveness and suitability for individualization are considered close to neutral, then the greatest room for improvement lies with error tolerance. This feedback has already been integrated into newer developments of the GLM, where frequent user guidance was added to the interface and messages inform the learning designer about errors or differing expectations of the software.

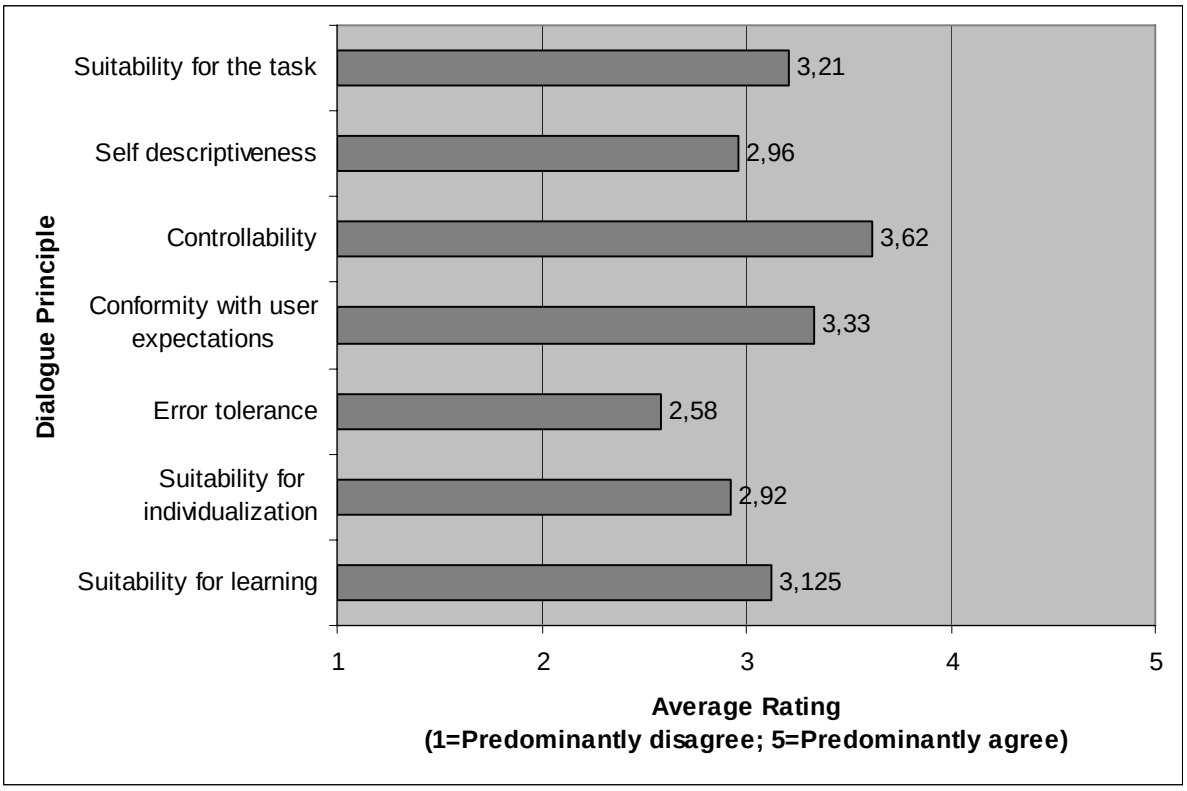

Figure 3: Evaluation results for GLM collected from industrial test bed partners according to ISO dialogue principles (adapted from Faltin, 2008).

Conclusion

The IMS LD specification provides the possibility to model complex learning settings. Its language, especially the language at level B, has been shown to be far from practitioners' language and design approaches. The analyses presented in this article formed the basis for developing easy access interfaces in IMS LD authoring software, which enables teaching practitioners to use advanced concepts in their learning designs without being cognizant of level B syntax and semantics. Behind the level B entry points, the software automatically

arranges the technical setup according to IMS LD from the patterns of steps that were aligned with each entry point.

The analyses further pointed out some deficiencies in regard to IMS LD, especially regarding the inability to react to spontaneous changes in the learning design once the unit of learning is under way, and the degree of difficulty when interpreting imsmanifest-XML-files for property uses. Also, if the responsibility for shaping the learning environment is increasingly shifted to the learners, the harder it gets to adequately express these learning designs in IMS LD as the specification caters to instructional design principles that foresee and pre-specify all elements.

Acknowledgements

This article was written in the context of the research and development integrated project PROLIX, which is co-funded by the European Commission under the Sixth Framework Programme "Information Society Technologies". The GLM mentioned in this paper is a development of the University of Vienna with major contributions from Philipp Prenner and Stefan Zander at the Multimedia Information Systems Group headed by Wolfgang Klas. We would like to thank our anonymous reviewers for their detailed feedback on an earlier draft that helped to improve this paper.

References

Barrett-Baxendale, M. (2007). A practitioner's view of Learning Design. *TENCompetence Open Workshop on Current Research on IMS Learning Design and Lifelong Competence Development Infrastructures* Retrieved February 21, 2008, from http://www.tencompetence.org/files/Barcelona/presentations/keynotes/keynote2.ppt

Beetham, H. (2004). Review: developing e-Learning Models for the JISC Practitioner Communities, Version 2.1. Retrieved March 13, 2007, from http://www.jisc.ac.uk/uploaded_documents/Review%20models.doc

Faltin, N. (2008). D9.3 PROLIX prototype version – pilot evaluation. Retrieved May 9, 2008, from http://www.prolixproject.org/index.php?option=com_remository&Itemid=45&func=fileinfo&id=107 (login required)

Griffiths, D., & Blat, J. (2005). The Role of Teachers in Editing and Authoring Units of Learning Using IMS Learning Design. *Advanced Technology for Learning, 2*(4).

Harvard Project Zero. Visible Thinking: Think-Pair-Share. Retrieved February 22, 2008, from http://www.pz.harvard.edu/vt/VisibleThinking_html_files/03_ThinkingRoutines/03d_UnderstandingRoutines/ThinkPairShare/ThinkPairShare_Routine.html

Heyer, S. (2007). *Preparing didactic models for the implementation in IMS Learning Design* (PROLIX Internal Document). Vienna: University of Vienna.

Heyer, S., Oberhuemer, P., Zander, S., & Prenner, P. (2007). Making Sense of IMS Learning Design Level B: from specification to intuitive modeling software. In E. Duval, R. Klamma & M. Wolpers (Eds.), *Lecture Notes in Computer Science 4753* (pp. 86-100). Berlin, Heidelberg: Springer.

Koper, R., Olivier, B., & Anderson, T. (Eds.). (2003a). *IMS Learning Design Best Practice and Implementation Guide*: IMS Global Learning Consortium, Inc.

Koper, R., Olivier, B., & Anderson, T. (Eds.). (2003b). *IMS Learning Design Information Model*: IMS Global Learning Consortium.

Koper, R., & Tattersall, C. (Eds.). (2005). *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*. Berlin, Heidelberg: Springer.

Nelson, L. M. (1999). Collaborative Problem Solving. In C. M. Reigeluth (Ed.), *Instructional-Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II, pp. 241-267). Mahwah, NJ: Lawrence Erlbaum.

Neumann, S., & Oberhuemer, P. (2008). *Supporting Instructors in Creating Standard Conformant Learning Designs: the Graphical Learning Modeller*. Paper presented at the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA), Vienna.

Sodhi, T., Miao, Y., Brouns, F., & Koper, R. (2007). Design Support for Non-Expert Authors in the Creation of Units of Learning -- a First Exploration. Retrieved February 21, 2008, from http://dspace.learningnetworks.org/handle/1820/984

van Es, R., & Koper, R. (2006). Testing the pedagogical expressiveness of IMS LD. *Journal of Educational Technology & Society*, *9*(1), 229-249.

W3C. (2006). Extensible Markup Language (XML) 1.1 (Second Edition). Retrieved May 7, 2008, from http://www.w3.org/TR/xml11/