Authoring of Units of Learning via Dialogue Systems

Dietmar Janetzko
School of Informatics, National College of Ireland,
Mayor Street, Dublin,
Ireland
Email: djanetzko@ncirl.ie

Abstract— **Detailed and expressive e-learning specifications like IMS LD [1] are necessary to create efficient e-learning applications. E-learning specifications that meet these demands tend to be long and detailed such that authoring e-learning content becomes a tedious and error-prone endeavour. This paper describes DBAT-LD (Dialogue-Based Authoring of Learning Designs) which is a document generating dialogue system (DGDS) that interacts with authors of units of learning and secures the content of the dialogue in an XML-based target format. In the case of IMS LD, DBAT-LD elicits an IMS LD compliant description of learning activities along with pedagogical support activities [2] and delivers a Unit of Learning package. DBAT-LD offers a lot of support (explanations, control questions, adaptation to the user) to ease authoring while strictly following the specification of IMS LD.**

Index Terms— **E-Learning, IMS LD, AIML, Dialogue Management, Conformance Testing**

I. Introduction

Which e-learning specification can be used to represent learning scenarios (lessons, courses, modules)? What are the criteria that should be met so that this description is sufficiently generic to be used by different kinds of e-learning applications? These are the questions that led to e-learning specifications like, e.g., IMS LD [1]–[3]. IMS LD can be used as a notation for different kind of pedagogical models, and it meets a number of more technical objectives (e.g., interoperability). Conceiving of e-learning specifications from this perspective is certainly important, if not indispensable when designing educational software. Willy nilly, however, this vantage point is closely related to the question how learning design knowledge to be used in online courses is captured in the first place. IMS LD remains silent about this question. Usually, text-based editors or graphical authoring tools are used for this purpose. While IMS LD editors certainly simplify the authoring procedure, they are still quite complex and provide only limited support with regard to reconstructing pedagogical knowledge. Dialogue-based authoring provides this kind of support by transforming the design activity into a storytelling process. Authoring e-learning content can be simplified by leading the user through the IMS LD specification and adapting to his understanding when necessary. The goal of this paper is to introduce a system and rationale for dialogue-based authoring of units of learning, i.e., IMS LD compliant specifications of learning scenarios (courses, modules, or lessons). Dialogue-based authoring of IMS provides a supplement or alternative to editor-based authoring. The top-level elements inside the learning-design element of the IMS LD specification (components, method and their component elements) are used as the running thread that guides an authoring interview with a human pedagogical expert on a learning scenario. In so doing, the dialogue-system prompts information needed to spell out a learning design, saves this information and delivers an IMS LD compliant output, i.e., a unit of learning in an IMS LD content package.

The paper is organized as follows: First, the editor-based and the dialogue-based way of authoring content for educational software are compared and contrasted. Secondly, discussing related work will show that previous approaches to authoring could not reach the level of generality of DBAT-LD. Thirdly, there is an outline of IMS LD and the multiple ways it supports DBAT-LD. Fourthly, language technology used for authoring, i.e., AIML (Artificial Intelligence Modeling Language) and the overall architecture if DBAT-LD is described. Finally, I will delineate skills DBAT-LD brings to bear when conducting authoring dialogues and present an example dialogue conducted with DBAT-LD.

II. Editor-Based and Dialogue-Based Authoring

When setting set up a learning design that is based on IMS LD course authors may choose between some sort of ASCII-editor, XML-editor or an IMS LD editor. The latter is a more specific authoring tool that is geared towards creating IMS LD compliant output. Examples of IMS LD editors are the Reload Learning Design Editor [4], LearningMapR [5], CopperAuthor [6], IMS LD graphic editor, [7] elive LD-suite [8], ASK LDT [9], or Mot+ [10]. Furthermore, there are several LD-related tools available like CopperCore or the Reload learning design player. These tools analyze learning design packages and test if they conform to the IMS LD specification.[1] While graphical editors provide a number of features to support the design of pedagogical scenarios, i.e., units of learning,

[1] A list of learning design tools currently available or under development can be found under [11].

they offer only limited support that helps to unravel pedagogical expertise such that it can be mapped to the complex specification of IMS LD.

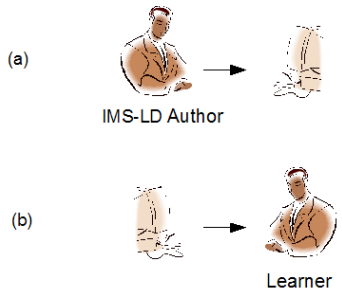

Figure 1. Dialogue-Based Authoring

Using dialogue systems like DBAT-LD constitutes a complementary or alternative road to authoring IMS LD. To a considerable degree, authoring online courses means eliciting and redescribing pedagogical knowledge in terms of an abstract notation with a technical flavor like IMS LD. Natural dialogues support this change of representation in a multiple of ways by

- explaining basic concept of the notation,
- providing examples that highlight the usage of basic concepts of the notation,
- prompting additional information required,
- redescribing utterances in a 'does-this mean-XY?' style,
- answering questions,
- scaffolding the overall process of redescription by rectifying misconceptions.

A natural language dialogue system, e.g., a chat bot system, allows only an approximation of human-human dialogues. Still, todays dialogue systems offer a range of features that make them an interesting tool for carrying out knowledge engineering and authoring tasks. Moreover, document generating dialogue systems (DGDS) like DBAT-LD secure the content of the dialogue by generating a document that can be used by other applications (e.g., e-learning systems).

The dialogue-based road outlined in this paper is targeted to authors of units of learning who have no or only a limited understanding of IMS LD. Supporting them via dialogue-based authoring may mean many things: Rectifying misconceptions, answering questions, providing examples, or reconstructing design knowledge that is possibly implicit. Once a unit of learning has been generated via dialogue-based authoring and the resulting specifications have passed a number of validation procedures and conformance testing it may be used by an IMS LD compliant player for e-learning (cf. Figure 1). Clearly, the dialogue-based road to authoring educational software has much in common with an editor-based road: Both approaches create an awareness of issues to be addressed in authoring (e.g., concerning the information required) and both assist the user editing one or several files that specify the unit of learning in IMS LD. Dialogue-based authoring may integrate graphics similar to those used in editor-based authoring. Vice versa, in editor-based authoring dialogues may be integrated. However, dialogue-based authoring has a greater potential to adapt to the user (e-learning author). For instance, DBAT-LD keeps a record of the dialogue history and adapts its performance accordingly. In addition, dialogue-based authoring is more suited to unravel pedagogical knowledge, e.g., by starting in a very general way and narrowing down the authoring dialogue to more specific levels of description.

III. Related Work

Collecting information and representing it in a structured way is one of the main reasons why dialogue systems are used [12]. In areas like text generation dialogue systems have been used to create the knowledge base that is used to generate texts [13]. It is therefore surprising that dialogue systems have been used rarely to author e-learning systems. The authors in [14] describe a set of tools dedicated to author natural language dialogue interface for intelligent tutoring systems (ITS). Is it shown that the authoring tools facilitate rapid development of natural language dialogue interfaces. In itself, however, the authoring tools do not seem to be dialogue-based. Moreover, the work lacks generality since the approach is applied only to Intelligent Tutoring Systems that have been developed by the authors themselves. In contrast, the work delineated in this paper follows a more general approach since the output generated by DBAT-LD can be used by any IMS LD compliant player.

IV. IMS Learning Design

IMS Learning Design (IMS LD) is an open e-learning specification released by the IMS Global Learning Consortium in 2003 [1]. IMS LD includes a binding specification that defines precisely how learning designs are represented in XML. IMS LD derived from two developments in the area of e-learning specifications: Firstly, IMS LD is based upon the Educational Modeling Language (EML), which was developed by Rob Koper [16] from the Open University of the Netherlands (OUNL). The motivation behind EML was to show that many, if not all pedagogical approaches can be described in a meta-language, the core of which accounts for various activities for one or several learner roles and staff roles in a certain order. IMS LD incorporates and extends the general approach of EML. It provides an elaborated notation to specify learning and teaching activities which is pedagogically neutral and implementation independent. Secondly, IMS LD integrates other IMS specifications like content packaging [17], meta data [18], and simple sequencing [19].

A. Levels of Implementation

IMS LD distinguishes three levels of specification, i.e., level A–C, each of which is mapped to separate XML schemas.

```
learning-design
    title
    learning-objectives
    prerequisites
    components
        roles
            learner*
            staff*
        properties*
        activities
            learning-activity*
                environment-ref*
                activity-description
            support-activity*
                activity-description
            activity-structure*
                learning-activity-ref
                support-activity-ref
        environments
            environment*
                learning objects*
                services*
                environment-ref*
    method
        play*
            act*
            role-parts*
            role-ref
          activity-ref
          activity-structure-ref
        conditions*
    metadata
```

Figure 2. Main Elements of IMS LD (Level A)

Learning Design Level A is the core of IMS LD and the majority of Units of Learning presented so far in work on IMS LD rely on level A. The level A specification of IMS LD provides a basic notation that can be used to express a large diversity of pedagogical approaches. The elements expressed on level A are shown in Fig 2. An asterisk * means that an element may occur more than once. The hierarchical structure of elements of the level A specification of IMS LD is taken as a guideline to carry out a dialogue-based authoring. Thus, DBAT-LD will ask for the title of the learning design, inquire about the requirements, find out which persons are involved in the learning activities, etc [1].

Learning Design Level B extends level A by adding properties, global elements, monitor services, and conditions.

Learning Design Level C adds notifications to level B. Notification are triggered by learning activity outcomes, which can make a new activity available for a role to perform.

There are several reasons that motivate a distinction of the IMS LD specification into different levels. Firstly, to avoid an overly complex specification that would include a number of mandatory elements and numerous optional elements. Secondly, compared to the more basis features of level A the features of level B and C enable a more elaborate behavior of the e-learning system, which may or may not be required. A distinction into levels facilitates a selection of these features when needed. Likewise, vendors or organizations that consider using IMS LD may decide to opt for particular levels. Thirdly, e-learning program development can be staged and geared towards particular objectives (e.g., adaptation).

The majority of work related to IMS LD (e.g., tool development) focuses on level A. More recently, however, more attention is given to level B since specification on this level holds the key for collaborative learning, adaptive learning and personalization, conditional text, new forms of assessment, or runtime tracking [20]. In contrast to most other e-learning specifications or standards IMS LD supports multi-learner activities and environments. It is generally agreed that IMS LD offers enough flexibility to describe and specify learning and teaching [21]. In the past, however, the uptake of IMS-ID has been hindered by the lack of user-friendly tools. With the advent of a number of tools (editors, players) this situation is beginning to change. More recent developments indicate an integration of IMS LD with well-known open source e-learning tools: LAMS (Learning Activity Management System) will integrate import/export feature using IMS LD Level A. Likewise, the open source course management system Moodle will be able to export in IMS LD or play an IMS LD packages. Vice versa, IMS LD tools will be able to import Moodle courses [22].

B. Units of Learning (UoL)

A number of resources (texts, HTML documents, figures to be presented to the learners, presentation slides etc.) may be required to specify learning and teaching activities. In addition, more technical information needs to be added (e.g., about namespaces). The IMS content packaging is used to bundle and cross-reference the variety of resource files [17]. Moreover, the information about the packaging rationale itself is specified in a manifest file. so that a player program like a learning management system can easily find the information needed. This bundle of information is usually provided in a compressed format (e.g., a zip file).

C. IMS LD and Dialogue Design

An XML-based e-learning specification like IMS LD offers excellent possibilities for designing a dialogue-based authoring tool. Since the elements of the specification define the information to be elicited DBAT-LD uses them as a guideline for the authoring process. This does not necessarily mean that a "fixed-pipeline" approach (a standard sequence of questions) is used throughout each dialogue. An element (e.g., `activities`) can be taken to mark a focal point in the dialogue. Starting from there, related themes like the single teaching activities, the environments of the activities, the support administered can be addressed in the dialogue all of which are component elements of the `activities` element. Thus, taking IMS LD as a guideline for dialogue managements contributes

to thematic coherence of the dialogue. Moreover, the specification provides a data structure that can be used for dialogue management. In DBAT-LD, this data structure is the basis for the dialogue management rationale chosen, i.e., a blackboard control architecture.

V. The Overall Architecture of DBAT-LD

DBAT-LD has been implemented in PHP. In particular, the PHP extension DOM (Document Object Model) has been extensively used. The DOM extension of PHP is an API which complies excellently with the language- and platform–neutral interface DOM standard of the World Wide Web Consortium (W3C) [23]. The DOM extension of PHP provides a number of functions that has been used in DBAT-LD to create, parse, and modify XML document instances.

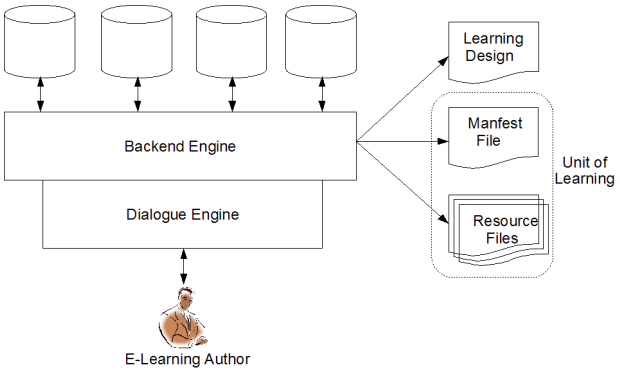

Figure 3. Overall Architecture of DBAT-LD

A. The Knowledge Bases

DBAT-LD makes uses of a modularized knowledge base that is used to for dialogue processing and management.

1) IMS LD: An instance of the IMS LD specification is built up and updated on the fly.

2) Background Knowledge: The IMS LD specification is augmented by WordNet [24] to detect and resolve synonymous concepts that are used in the dialogue.

3) Dialogue History & User Model: The dialogue history is kept to allow memory-based processing within a dialogue. Adaption to the user makes use of the dialogue history.

4) Examples of Units of Learning: Examples are represented that are shown to the user upon request.

B. The Dialogue Engine

The dialogue engine used by DBAT-LD is based on AIML (Artificial Intelligence Markup Language). AIML is a XML-based pattern language developed by Dr. Richard Wallace and the Alicebot free software community [13-15]. One of the chief motivations that lead to the development of AIML was to design a simple language for machine-based mixed-initiative dialogues. AIML is applied in various applications (e.g., computer games). Among others, it has been used to design tutorial dialogues [25], [26].

Processing AIML is an example of shallow natural language processing (no grammatical parsing). For this reason, AIML-based natural language processing systems are usually called chat bot systems. The principal elements of the dialogue engine used are

- AIML Interpreter
- AIML Code
- Dialogue Management Module
- Agents

Next is a description of the components of the dialogue engine of DBAT-LD.

1) AIML Interpreter: An AIML interpreter is required to render the AIML code. A number of open source AIML interpreters are available each of which provides particular features (e.g., interfaces to particular knowledge bases). The AIML interpreter used by DBAT-LD is ProgramE since it is written in PHP and thus ties in well with the remainder of the system.

2) AIML Code: The AIML code is a set of adjacency pairs [27] like initiation/response or question/answer. In addition, AIML code for preprocessing the input has been used. Preprocessing was required to increase the systems robustness with regard to different wordings of comparable user utterances.

3) Dialogue Management Module: Dialogue management is a key issue in all dialogue systems. In DBAT-LD the dialogue management module controls dialogue planning and thus the overall flow of the dialogue. The dialogue module makes use of a blackboard architecture [28] that acts as a common data structure for specifying conditions and goals that relate to the agents. For instance the blackboard may list the goals

```
goal = (title, learning-objectives,
prerequisites, components, method)
```

Enabling conditions are used to control the activity of the dialogue engine. For the dialogue engine to become active (by contributing to the dialogue) a set of preconditions have to be met. Once the goal is accomplished, it will disappear from the blackboard. In addition, goal accomplishment may affect the fulfillment of enabling conditions. In this way a pro-active behavior of the dialogue system can be guaranteed without relying on a predetermined sequence of goal accomplishment. DBAT-LD is a mixed-initiative dialogue system and thus major tasks to be achieved in dialogue management in DBAT-LD can be described as follows

- Managing incoming utterances of the user by passing the request to the component of the dialogue system that is suited best to give an answer.

- Taking the initiative if needed. DBAT-LD takes the initiative if an issue is settled, if an issue necessitates follow-up questions of if the user does not take the initiative (remains silent).
- Management of themes by staying with them until all open questions are answered or by coming back to a theme.
- adapting to the user by assessing his level of expertise and performing accordingly.

4) Agents: The dialogue management in DBAT-LD is achieved via agents or dialogue specialists. To the user (e.g., the educational expert), the dialogue system may appear to be a homogeneous entity. However, the dialogue management and thus the authoring of IMS LD compliant documents is based on the joint activity of a community of agents, i.e., conversational specialists that focus on a particular topic of the authoring process. The agents are part of the dialogue management module which in turn makes use of a blackboard control structure. A blackboard is a common data structure all agents may access for reading and writing [28]. The agents of the dialogue system reflect the organization of Learning Design. For each of the component elements of the root element of a unit of learning, i.e., the `learning-design` element, there are agents that guide the conversation on title, learning-objectives, prerequisites, components, method and meta-data. We may as well say that each agent addresses a particular goal in the authoring process. For instance, the learning-objectives agent will ask questions about the objectives of the unit of learning talked about, etc. Each agent reconstructs chunks of learning design knowledge that fall into its scope and saves them. In this way, the community of all agents contributes to producing an IMS LD compliant output file.

The motivation for using agents to implement dialogue management is twofold: Firstly, agents reflect the modular structure of the dialogue knowledge-base. Secondly, when integrated in a blackboard control structure agents support flexible dialogue patterns. Thus, instead of using a predetermined sequence of questions ('fixed-pipeline approach') the dialogue flexibly follows topic changes if suggested by the user. The dialogue will, however, come back and address major dialogue goals and will finally generate an IMS LD compliant document. The agents have a number of features the most striking of which are their greediness, stickiness and responsiveness.

Greediness of Agents. Designed to be greedy, dialogue specialists do want to contribute to the dialogue. However, before agents may perform, their enabling conditions need to be fulfilled. The same is true with respect to utterances of one agent. It does not seem reasonable, for instance, to start authoring a learning unit by addressing the method at the very start. Likewise, there is no point in discussing support activities if other aspects of the method have not been dealt with. Once the enabling conditions of a dialogue specialist are fulfilled, it may play an active or passive role in the dialogue. In so doing, changes are introduced to the blackboard that will have enabling or disabling effects on other agents. When the learning-objectives agent has finished its job, it will express this by leaving a message on the blackboard. This, in turn, may or may not enable other agents to take control of the dialogue (by focusing on the prerequisites of the unit of learning under discussion). Depending on the enabling conditions, the performance sequence of dialogue specialists, i.e., the topics of a dialogue may be linear, partially ordered or devoid of any pre-specified order.

Stickiness of Agents. Apart from being greedy dialogue specialists are also sticky. This means that once a dialogue specialist is performing, it will tend to stay with this topic. Stickiness of dialogue specialists has been added to the features of the dialogue system to prevent it from becoming mercurial, i.e. jumping across various topics. Stickiness is compatible with each dialogue style (e.g., system-initiative, user-initiative, mixed-initiative), but most relevant if the dialogue system steers the dialogue (system-initiative). Stickiness of the dialogue specialists is accomplished by features of the AIML-syntax (in particular the `TOPIC`-tag) of AIML.

Responsiveness of Agents. Though being sticky, agents are in fact responsive. Responsiveness of dialogue specialists means two things: First, the system will answer questions, provide clarifications or even change the overall dialogue agenda upon request of user. Second, the system will consider the user model when deciding whether the control should be passed over to a particular content-related agent. For instance, if the user model tells that the topic of learning-objectives has already been presented to the user (i.e., the corresponding agent has already done its job), there is little reason to pass over control to the same agent again. At least this is not done without explicitly asking the user whether or not to do so.

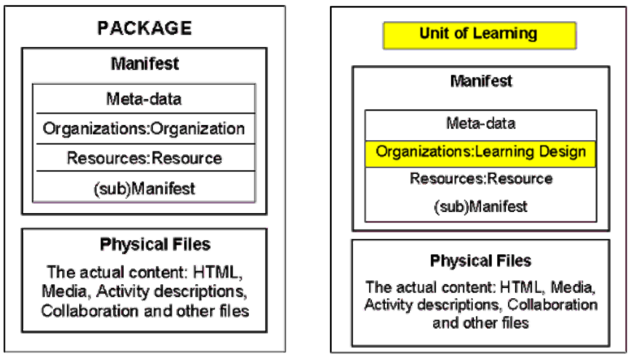

Figure 4. Content Package and the embedded Learning-Design
The left figure presents the structure of the IMS LD compliant content package as it is created by DBAT-LD. The right figure sets out how the IMS Learning Design (elicited via the authoring dialogue) is integrated with the IMS Content Package. as a component element of the `organization` element [1]

C. The Backend Engine.

The backend engine carries out a number of authoring tasks that are shielded away from the e-learning author.

1) Generation of a manifest file: The manifest file `imsmanifest.xml` is the central element of an IMS LD unit of learning. Information that is elicited in the authoring dialogue, i.e, the specification of the learning design as well as resources files need to be integrated with the manifest file. This means, information about a learning design is actually included in the learning design element. Information about resources (e.g., files used in a course that is described via IMS LD) is usually included in separate files that need to be referenced by resource elements in the manifest file.

Moreover, the manifest file contains technical information (e.g., about name spaces) that are not elicited in the authoring dialogue. The backend engine addresses all issues mentioned above. It gleans information collected in the authoring dialogue and integrates it into the manifest file. If needed, it retrieves information from the knowledge base that includes technical information about IMS LD and writes it to the manifest file.

2) Conformance Testing: If e-learning content and software is intended to be reused then it needs to comply with standards and specifications that have been devised to support this objective. Quite often, however, it is not clear whether standards and specifications are actually met. For instance, standards and specifications are not set in stone but exist in different versions. E-learning content may be drawn from various sources some of which may or may not comply with a particular e-learning standard or specification. Thus, before a newly generated unit of learning is packed conformance testing [29] is required, which includes

- non validating parsing to ensure the basic syntactical correctness of the manifest file,
- validating parsing, i.e. a test of the manifest file against the IMS LD schema (for level A) of the IMS LD specification,
- check of the availability and correct reference of the resource files specified in the manifest file.

3) Generating a Learning Design Package: Once the manifest file has been created, and the quality control was passed successfully, the learning design package is generated. To this purpose the backend engine packs the manifest file and the resource files referenced in the manifest file into a zipped file the structure of which is shown in Figure 4).

VI. Features of Dialogue-Based Authoring

Dialogue-based authoring targeted to setting up learning designs have a specific profile. Seen from a more general point of view, the profile of dialogue-based authoring can be described as being mixed initiative and open: Firstly, the set-up of the dialogue system is geared towards eliciting or reconstructing knowledge as specified by IMS LD. As a consequence, for most part of the dialogue the initiative (e.g., addressing and pursuing a particular topic by asking questions) is mainly on the system side. Clearly, however, the user may also take the initiative and ask questions. User questions may, for instance, be provoked by concepts DBAT-LD has employed in questions. Secondly, conversations in dialogue-based authoring are conducted to elicit, reconstruct and redescribe pedagogical knowledge. In DBAT-LD unraveling pedagogical knowledge is done via a number of discourse mechanisms like paraphrasing used utterances and requesting a user feedback, providing examples and asking for the reason or motivations of user decisions. Thirdly, by its very nature authoring is an open modeling activity and thus full transparency of the IMS LD compliant description generated is indispensable. At various points in time during an authoring dialogue the user may inspect the IMS LD compliant description of the learning unit (cf. Table II).

Dialogue skills of DBAT-LD can be specified on the levels of words, sentence or the overall conversation or discourse (cf. Table II). Performance of each of the skills

TABLE I.
Dialogue Skills of DBAT-LD

Feature	Performance
Explaining Concepts	DBAT-LD explains concept autonomously or when asked to do so. In this way, the system supports the user who is unfamiliar with the lingo of IMS LD.
Example (User):	`Please give me an example of a support activity`
Asking for a Rephrase	If DBAT-LD is not able to process the utterance it asks the user to rephrase it.
Example (System):	`I did not quite understand what kind of learning-activity is required. Could you please rephrase your description of a learning-activity? A one-word description would be most welcome.`
Pumping for more information	Some elements of IMS LD may contain a sequence of component elements. DBAT-LD needs to find out if more information is required.
Example (System):	`I understood that Study-Resources and Clarify-Problem are learning-activities. Are there any other learning-activities?`
Requesting Confirmation	Once DBAT-LD has completed an Element it asks for a confirmation.
Example (System):	`It looks like there are basically two participants or rather roles involved in this unit of learning: a learner and a tutor. Is that correct?`
Paraphrasing User Utterances	Paraphrasing user utterances and asking for confirmation is an essential part of reconstructing learning design knowledge.
Example (System):	`Is it correct that in this act the support activity is to provide a feedback (provide-feedback)?`
Subtopic Management	The system uses a topic stack that supports subtopic management. Thus, the system will come back to a topic that has been addressed shortly but not exhaustively.
Example (System):	`Let us come back to the participants involved in this unit of learning.`

TABLE II.
DIALOGUE SKILLS OF DBAT-LD

Level	Skill	Pro-active	When Asked
Word	Providing Explanations	•	•
	Asking for Explanations	•	
	Relationship Management		
Sentence	Repeating last Utterance	•	
	Pumping for More Information	•	
	Paraphrasing User Utterances	•	
Discourse	Sub-dialogue Management	•	
	Summing Up	•	•
	Presenting Examples	•	•
	Disclosure of Content		•

my be triggered autonomously or heteronomously (when asked). For instance, an autonomous performance of the skill *Explaining Concepts* means that the system will come up with a explanation of a particular concept without being asked to do so. A heteronomous performance of the same skill simply means that the system will provide an explanation when the user ask for it. In this way, DBAT-LD supports the user who is unfamiliar with the lingo of IMS LD. The other dialogue skills of DBAT-LD can be classified accordingly.

VII. CONCLUSION

E-learning specifications compel and facilitate knowledge elicitation for e-learning. DBAT-LD reacts to this new challenges and possibilities by making dialogues a central part of the authoring process. On the one hand, DBAT-LD can be compared to editor-based approaches to authoring Units of Learning. On the other hand, DBAT-LD can be discussed with respect to dialogue-based authoring. With regard to editor-based approaches to authoring IMS LD compliant documents DBAT-LD helps the user to come to terms with the complex specification of IMS LD. This is achieved by applying a number of dialogue feature to the authoring process (e.g., sub-dialogue management, paraphrasing use utterances and adapating to the level of expertise of the user. With regard to authoring e-learning via dialogues DBAT-LD shows that by virtue of using a well-defined specification like IMS LD the generality of the output is increased. In addition, new forms of authoring (authoring in teams and re-authoring of previous units of learning are supported on the basis of IMS LD.

Generality of the Output generated by DBAT-LD.. DBAT LD supports dialogue-based authoring of elearning system content that – in principle – can be used in a large number of e-learning systems. The generality of DBAT LD has been made possible by the e-learning specification used. The contribution of IMS LD can be summarized as follows. Firstly, the XML basis of IMS LD can be used to capture and represent the knowledge elicited. Secondly, a specification like IMS LD supports a focused authoring procedure. Thirdly, IMS LD provides a common data structure which in turn can be used for more advanced dialogue management rationales. Thirdly, the IMS LD specification can be used as a yardstick to check the completeness of the knowledge elicited. Fourthly, the IMS LD specification can be used to support authoring of multiple authors or a re-authoring of previous instances Finally, IMS LD provides a framework for target format to represent and save Units of Learning.

Generality of the Approach taken by DBAT-LD The current version of DBAT-LD is implemented in AIML. However, since the well-defined generic representation, i.e., IMS LD, is kept separate it would be relatively easy to replace the AIML-based dialogue engine by a different set of tools (e.g., by TRINDIKIT, [15]) while retaining the core representation. Likewise, dialogue-based authoring similar to the way DBAT-LD proceeds can be carried out on the basis of other specifications.

ACKNOWLEDGEMENT

I like to thank the anonymous reviewers for the constructive and well thought out questions and comments.

REFERENCES

[1] IMS Global Learning Consortium, "IMS Learning Design Information Model. Version 1.0 Final Specification," Retrieved January 15, 2007 from http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html, 2003.

[2] R. Koper and B. Olivier, "Representing the learning design of units of learning," *Educational Technology & Society*, vol. 7, no. 3, pp. 97–111, 2004.

[3] B. Olivier and C. Tattersall, "The Learning Design specification," in *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*, R. Koper and C. Tattersall, Eds. Berlin: Springer, 2005, pp. 21–40.

[4] C. Milligan, P. Beauvoir, and P. Sharples, "The reload learning design tools," *Journal of Interactive Media in Education*, vol. August, 2005.

[5] D. Buzza, Richards, D. L. Bean, K. Harrigan, and T. Carey, "LearningMapR: A Prototype tool for Creating IMS-LD Compliant Units of Learning," *Journal of Interactive Media Research in Education*, vol. August, 2003.

[6] W. van d.er Vegt and R. Koper, "CopperAuthor v1. 6," Retrieved January 15, 2007 from http://dspace.ou.nl/handle/1820/592, 2006.

[7] E. G. Pacurar, P. Trigano, and S. Alupoaie, "A QTI editor integrated into the netUniversit web portal using IMS LD," *Journal of Interactive Media in Education*, vol. September, 2005.

[8] "elive LD-suite," Retrieved January 15, 2007 from http://www.elive-ld.com/content/e47/e649/index_eng.html, 2006.

[9] D. Sampson, P. Karampiperis, and P. Zervas, "ASK-LDT: A Web-Based Learning Scenarios Authoring Environment Based on IMS Learning Design," *International Journal on Advanced Technology for Learning*, pp. 207–215, 2005.

[10] G. Paquette, d. I., and M. Léonard, "An instructional engineering method and tool for design of units of learning," in *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*, R. Koper and C. Tattersall, Eds. Berlin: Springer, 2005, pp. 161–184.

[11] Unfold Project, "Learning design tools currently available or under development," http://www.unfold-project.net:8085/UNFOLD/general_resources_folder/tools/currenttools.

[12] M. Witbrock, D. Baxter, J. Curtis, D. Schneider, R. Kahlert, P. Miraglia, P. Wagner, K. Panton, G. Matthews, and A. Vizedom, "An Interactive Dialogue System for Knowledge Acquisition in CYC," *International Joint Conference on Artidicial Intelligence (IJCAI-03). Proceedings of the Workshop on Mixed-Initiative Intelligent Systems*, 2003.

[13] A. Knott and N. Wright, "A dialogue-based knowledge authoring system for text generation," *AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue, Stanford, CA*, 2003.

[14] P. Jordan, C. Rose, and K. VanLehn, "Tools for authoring tutorial dialogue knowledge," Washington, DC, pp. 222–233, 2001. [Online]. Available: citeseer.ist.psu.edu/jordan01tools.html

[15] S. Larsson and D. Traum, "Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit," *Natural Language Engineering Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering*, vol. 6, pp. 323–340, 2000.

[16] R. Koper, "Modeling units of study from a pedagogical perspective: the pedagogical meta-model behind EML," Retrieved January 15, 2007 from http://dspace.ou.nl/handle/1820/36, 2001.

[17] IMS Global Learning Consortium, "IMS Content Packaging Specification, Version 1.1.3 ," Retrieved January 15, 2007 from http://www.imsglobal.org/content/packaging/, 2003.

[18] ——, "IMS Learning Resource Meta-data Specification, Version 1.3 – Final Specification," Retrieved January 15, 2007 from http://www.imsglobal.org/metadata/index.html, 2002.

[19] ——, "IMS Simple Sequencing Specification, Version 1.3 – Final Specification," Retrieved January 15, 2007 from http://www.imsglobal.org/simplesequencing/index.html, 2003.

[20] R. Koper and D. Burgos, "Developing advanced units of learning using IMS Learning Design level B," *International Journal on Advanced Technology for Learning*, vol. 2, no. 4, pp. 252–259, 2005.

[21] C. Knight, D. Gasevic, and G. Richards, "An ontology-based framework for bridging learning design and learning content," *Educational Technology & Society*, vol. 9, no. 1, pp. 23–37, 2006. [Online]. Available: http://www.ifets.info/journals/9_1/4.pdf

[22] Burgos, D. and Tattersall, C. and Dougiamas, M., Vogten, H. and Koper, E. J. R., "Mapping IMS Learning Design and Moodle. A First Understanding," in *Proceedings of Simposo Internacional de Informática Educativa (SIIE06), León, Spain: IEEE Technical Committee on Learning Technology*, 2003.

[23] World Wide Web Consortium, "Document Object Model (DOM)," Retrieved January 18th, 2007, from http://www.w3.org/DOM/, 1998.

[24] C. Fellbaum, *WordNet: an electronic lexical database*. Cambridge, MA: MIT Press, 1998.

[25] C. Lee, S. Han, and Y. Kim, "Educational application of dialogue system to support e-learning," in *World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA)*. Chesapeake, VA: AACE, 2002, pp. 984–989.

[26] O. De Pietro and G. Frontera, "Tutorbot: An application AIML based for web-learning," in *7th IASTED International Conference on Computers and Advanced Technology in Education (CATE 2004)*, 2004.

[27] S. C. Levinson, *Pragmatics*. Cambridge: Cambridge University Press, 1983.

[28] A. Kerminen and K. Jokinen, "Distributed dialogue management in a blackboard architecture," in *Proceedings of the EACL Workshop Dialogue Systems: interaction, adaptation and styles of management, Budapest, Hungary*, 2003, pp. 55–66.

[29] R. Nadolski, O. O'Neill, W. v. d. Vegt, and R. Koper, "Conformance testing, the elixer within the chain for learning scenarios and objects," in *ICALT '06: Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, L. T. Kinshuk, R. Koper, P. Kommers, P. Kirschner, D. G. Sampson, and D. W., Eds. Washington, DC, USA: IEEE Computer Society, 2006, pp. 363–365.

Dietmar Janetzko received his PhD degree in psychology from Freiburg University, Germany in 1996. In 2006, he acquired a PhD degree in learning sciences from the Technical University Kaiserslautern, Germany. He is currently a lecturer at the school of informatics of the National College of Ireland, Dublin, Ireland. His current research interests include tutorial dialogue systems, probabilistic models, assessment, constructivism, and Item Response Theory.