Education and Information Technologies 10:1/2, 81–108, 2005.

A Conceptual View of Web-Based E-Learning Systems

KLAUS-DIETER SCHEWE
Department of Information Systems & Information Science Research Centre, Massey University, Private Bag 11 222, Palmerston North, New Zealand
E-mail: k.d.schewe@massey.ac.nz

BERNHARD THALHEIM and ALEKSANDER BINEMANN-ZDANOWICZ
Department of Computer Science and Applied Mathematics, Christian Albrechts University Kiel, Olshausenstr. 40, D-24098 Kiel, Germany
E-mail: thalheim@is.informatik.uni-kiel.de
E-mail: binemann@is.informatik.uni-kiel.de

ROLAND KASCHEK
Department of Information Systems & Information Science Research Centre, Massey University, Private Bag 11 222, Palmerston North, New Zealand
E-mail: r.h.kaschek@massey.ac.nz

THOMAS KUSS
Center for Cognitive Science, Freiburg University, Germany
E-mail: thomas@cognition.iig.uni-freiburg.de

BERND TSCHIEDEL
Department of Computer Science, Brandenburgian Technical University at Cottbus, Universitätsplatz 1, 03044 Cottbus, Germany
E-mail: tschied@informatik.tu-cottbus.de

Abstract

Starting from a general framework for web-based e-learning systems that is based on an abstraction layer model, this paper presents a conceptual modelling approach, which captures the modelling of learners, the modelling of courses, the personalisation of courses, and the management of data in e-learning systems. Courses are modelled by outline graphs, which are further refined by some form of process algebra. The linguistic analysis of word fields referring to an application domain helps to set up these course outlines. Learners are modelled by classifying value combinations for their characteristic properties. Each learner type gives rise to intentions as well as rights and obligations in using a learning system. Intentions can be formalised as postconditions, while rights and obligations lead to deontic constraints. The intentions can be used for the personalisation of the learning system to a learner type. Finally, the management of data in an e-learning system is approached on two different levels dealing with the content of individual learning units and the integrated content of the whole system, respectively. This leads to supporting databases and views defined on them.

Keywords: e-learning systems, course modelling, learner modelling, personalisation, content management

1. Introduction

More and more education institutions aim at providing e-learning systems, and more and more people take advantage of these offers as a chance to advance their knowledge. It is envisioned that there is a huge market potential for learning technologies. Therefore, there is an increasing demand for adequate e-learning systems of highest standards, and a requirement for powerful techniques supporting agile and consistent learning. Expectations, chances and quality requirements are high. In particular, it still has to be proven that e-learning systems can provide the same (or nearly the same) teaching quality as contact classes with human teachers and adequate resources. In order to meet these high expectations professional design and development support for e-learning systems will be a crucial success factor.

The challenge in e-learning addresses at least two parts: the dissemination of information and the concept of teaching. In this article we concentrate on the first aspect, which reduces the problem to

- deciding, which information is to be presented in which form;
- enabling learners to chart individual routes through this information in accordance to their preferred learning style;
- adapting the information to different learner types.

We acknowledge that the concept of teaching is wider. For instance, didactics is more than outlining the content of a course. There is a considerable discussion in education regarding this wider perspective. However, learning technology has not yet reached a stage, where all the desiderata of e-learning systems can be supported. Our work emphasises the conceptual design of feasible systems and thus will demonstrate what can already be done with available technology, which is already more than is available in many systems on offer.

Our research interest concerns mainly those e-learning systems that are offered via the world-wide web. These e-learning systems can be considered as specific web-based information systems with a focus on the provision of knowledge to learners. In analogy to the B2C and B2B patterns that are well known in electronic commerce, we may use a pattern T2S for e-learning systems, where T represents a teacher as the service provider and S represents a student as the service consumer. More precisely, following the classification pattern in Thalheim and Düsterhöft (2000, 2001), we have a patterm $\mathrm{T}^k2\mathrm{S}^l$ with k standing for knowledge as the service that is provided, and l standing for learning as the major activity supported by the system.

In this article we start from a general framework for web-based e-learning systems that is based on an abstraction layer model (Kaschek *et al.*, 2003, 2004b). We present a conceptual modelling approach, which captures the modelling of courses, the modelling of learners, the personalisation of courses, and the management of data in e-learning systems. This exceeds the capability of approaches in Atzeni *et al.* (1998), Ceri *et al.* (2003) and Conallen (2003), which are based on a hypertext-extension of traditional development methods for data-intensive systems, and approaches such as Van Duyne *et al.* (2002), which concentrate only on the presentation aspect.

Many learning systems that are currently offered are based on curriculum sequencing, where the learner has to follow a well-defined sequence of learning steps. To some extent this follows the principles of didactic preparation as described in detail in Kerres (2001). This is not always adequate, as learning should be better based on active request, i.e. e-learning systems should support self-organised learning on demand. Systems should restrain the learners as little as possible, and offer instead as much support as possible to support their individual learning styles. Consequently, it is important to anticipate the behaviour of learners and to design systems according to their needs. This includes outlining courses in such a way that the sequence of learning units, and the style of presentation is personalised to the type of learner. More abstractly speaking, for each learner type we have to anticipate how they will navigate through the system. Each possible sequence of learning units followed by a learner corresponds to a particular course outline, so the most challenging problem is to determine these sequences and to describe them in an abstract and integrated way. We will show that this can be modelled by some form of process algebra.

This approach is similar to storyboarding in web information systems (Binemann-Zdanowicz *et al.*, 2004; Kaschek *et al.*, 2004b; Schewe and Thalheim, 2001), which has been applied to e-commerce (Binemann-Zdanowicz *et al.*, 2003a; Schewe *et al.*, 2002; Thalheim *et al.*, 2003), e-learning (Binemann-Zdanowicz *et al.*, 2003b; Jantke *et al.*, 2003; Rostanin *et al.*, 2002), and information services (Feyer *et al.*, 2000). The `SiteLang` process algebra has been introduced in Thalheim and Düsterhöft (2001). A short description of outline graphs also appeared in Binemann-Zdanowicz *et al.* (2004).

We also address the problem how to discover the best outline graphs. We observe that an atomic learning unit is always centered around a single learner activity by the learner, and this activity can be described by a verb. Therefore, the idea is to analyse verbs. Fortunately, the number of verbs in a language such as English is relatively small, so it is not an intractable approach. In particular, we suggest to analyse the *word fields* of verbs, which combine aspects of morphology, i.e. the forms of the written or spoken word, phonology, i.e. the sound of the spoken word, syntax, i.e. the construction of sentences using the word forms, semantics, i.e. the meaning(s) of the word, and pragmatics, i.e. the usage of the word in written or spoken language. This gives rise to determining the data needed to perform the activity, the guidance how to perform the activity, and the explanation of the effect of the activity.

The use of linguistic analysis for the design of web information systems has already been proposed in Ravenscroft and Matheson (2001) and Thalheim and Düsterhöft (2004) based on computational linguistics (Hausser, 2001). In addition, the use of metaphors has been proposed in general in Thalheim and Düsterhöft (2000) and investigated in Schewe *et al.* (2002) for applications in e-banking, and in Rostanin *et al.* (2002) for e-learning.

Furthermore, the quality of e-learning systems crucially depends on the designers' understanding of the learners and their needs. Therefore, it is necessary to first obtain an idea of the expected learners. This may lead to certain learner profiles. Such profiles may be determined by the different goals of the learners, their different intentions, their different behaviour, their information needs, their levels of required support, etc. However, we go further and discuss dimensions that are closer to the particular interest of modelling learners instead of users of arbitrary web information systems or banking customers. However,

similar to the existing work found in the literature we will obtain a *learner space*. We then discuss how points in this space can be combined into *learner types*.

Our approach adopts the whole-person approach from Martinez (2001) and the work on user profiling by Kaschek *et al.* (2004b) as far as we base it on learner dimensions. Also Hübscher (2001), Merrill (1983) and ONTO-LOGGING Consortium (2002) discuss learner modelling. A short description of learner modelling appeared in Kaschek *et al.* (2004a).

Furthermore, each learner type will give rise to intentions as well as rights and obligations in using a learning system. Intentions can be formalised as postconditions, while rights and obligations lead to deontic constraints. The intentions can be used for the personalisation of the learning system to a learner type. We will discuss personalisation based on subgraphs, but there are other techniques such as providing different granularities through splitting and merging units. Reduction to subgraphs can be formally supported by Kleene algebras with tests (Kozen, 1997). Splitting of units based on cohesion pre-orders was investigated in Feyer *et al.* (2000) and briefly described in Binemann-Zdanowicz *et al.* (2004).

Finally, we address the problems associated with the system's content, and partly with its functionality. We assume that learner types have been determined, and a course outline consisting of learning units and navigation links between them has been set up. While these already determine the functionality in a rough way, our interest is looking at the system from a more systems-oriented perspective and asking how we can support the learning units. As e-learning systems are data-intensive systems, we adopt the approach by Feyer *et al.* (2000), which approaches the data management problem in web information systems in an integrated way. According to this we obtain two levels of data management: a global level that is captured by a global database schema, and a local level that is captured by views, which represent the data content of the learning units.For both levels conceptual data models such as the Higher-Order Entity-Relationship model (Thalheim, 2000) are useful.

Among the more specific literature on e-learning, especially web-based e-learning systems very little attention is paid to the aspect of content modeling. Bergstedt *et al.* (2003) study similarities between content management systems (CMSs) and e-learning systems and conclude that using CMSs in e-learning would improve the quality of e-learning systems. The work in Qu and Nejdl (2003) and Sessink *et al.* (2003) investigates the metadata standard SCORM. Sessink *et al.* (2003) identify the need for supporting databases that are not in the SCORM 1.3 standard. Qu and Nejdl (2003) use XML to encompass incompatibilities between RDF and SCORM metadata. Mohan and Brooks (2003) address the problem of discovering learning objects from the semantic web and argue that this will improve learning. However, this depends on the semantic understanding of the web, which despite the merits of the research efforts in this area is still beyond reality. A short description of our approach to data modelling in e-learning systems appeared in Rostanin *et al.* (2004).

We present our general framework in Section 2. Then we address the modelling of courses via outline graphs in Section 3, followed by a discussion of linguistic analysis in Section 4 as a means for setting up adequate course outlines. Section 5 is then devoted to the classification of learners following the whole-person approach from Martinez (2001). In addition, we discuss intentions, rights and obligations that are associated with learner types. These link the learner types to the course outlines. Furthermore, intentions of learners can

be used to personalise learning system to the needs of the learners. This will be described in Section 6. Finally, in Section 7 we describe how the data in the various learning units can be modelled by using a two-level approach. We conclude with a brief summary in Section 8.

2. General Design Considerations

E-Learning Systems are large web-based information systems, and as such they share a lot of similarities with web information systems in general. Consequently, systems can be largely designed along the same principles that apply to traditional information systems. However, a particular focus has to be put onto the fact that the system will be web-based, and that its major purpose is to support learning.

This leads to important questions such as "Who are the learners?", "Which learner intentions and behaviour shall be supported?", "Which technical devices will be used by the learners?", etc. We will now look at these questions into more detail focussing on five different aspects: purpose, usage, content, functionality and presentation.

2.1. Aspects of e-learning systems

The purpose aspect is a very general one centered around a mission statement for the system. The primary question is: what is the purpose of the system? In e-learning systems the major purpose may be to provide learning material to students including useful hints and links to supplementary literature. A second related question is this: Are there several minor purposes as well? In commercial e-learning systems a minor purpose may be to attract students to book further courses, in which case the system would be a mixture of an e-learning and an e-commerce system. A third question associated with purpose concerns their time-scale. Some of the intentions may be long-term others short-term.

Once some clarity with respect to the purpose of the e-learning system has been obtained, the question arises by whom and how the system will be used. As a web-based system it is usually open, so it is important to anticipate the behaviour of the learners. Therefore, it is necessary to first obtain an idea of the expected learners. This may lead to certain learner profiles. Such profiles may be determined by the different intentions of the learners, their different behaviour, their information needs, their levels of required support, etc.

So, the activity of learner profiling will lead to a list of profiles of expected learners who are to be supported by the system. This influences the content of the system's pages, their logical organisation, the enabled navigation links between these pages, and maybe even their presentation. More abstractly speaking, for each learner profile we have to anticipate how they will navigate through the system. Each possible sequence of pages followed by a learner corresponds to a particular *course outline*, so the most challenging problem is to determine these sequences and to describe them in an abstract and integrated way.

The usage of an e-learning system depends on whether the control of the learning process is left to the learner or the system. In both cases, however, it is assumed that the learners are willing to learn and match the required prerequisites. Learners normally enter the system more than once continuing a specific learning programme. This requires some authentication

mechanism, especially if the learning progress is controlled by the system. Quality criteria are set by the teaching quality, and in the end, by the increase of knowledge on the side of the learners.

The content aspect is central to the development of the system, as it concerns the question: "Which information should be provided?", which is coupled with the problem of designing an adequate database. However, the organisation of data that is presented to the learner via the web-interface is significantly different from the organisation of data in a database. So, organising the data content of the system means to investigate the decomposition, structuring and classification of data in such a way that the course outline(s) can be adequately supported.

Thus, modelling the content of a system has to be addressed on at least two levels: a logical level leading to databases, and a conceptual level leading to the content of pages. Both levels have to be linked together. Furthermore, in both cases abstraction mechanisms should be used. While such abstraction mechanisms are established in the area of databases, they are still a matter of research for web-based systems and in particular for e-learning systems.

Modelling content must take into account that information must be presented in different ways to different learners. This depends on the learner profile, the communication channel, and the available devices. Modelling content has to provide mechanisms to tailor the content automatically according to these parameters.

The functionality aspect is coupled with the question, whether the e-learning system should be passive or active. A passive system would only allow a learner to navigate through the pages without any activity. In these cases the major problem associated with functionality is to set up an adequate navigation structure.

In an active system, however, information would also be required from the learner. From a conceptual point of view, the main purpose of functionality modelling is to identify functions that are available to support the activities of the learners, which were identified in the course outline. Such functions can be system-specific functions in order to process learner input or general support functions.

The functionality of e-learning systems mainly supports the navigation through the learning material. In contrast to other types of web-based systems, this navigation is a long-term process with usually many interruptions. More sophisticated systems would provide system-driven repetition and feedback. Also, personal information needs can be supported by providing an interface to e-mail.

Finally, the presentation dimension concerns the realisation by web pages. This depends on the support of technical end-devices such as computer screens, television, cell phones, etc. and set layout preferences. This is partly done in accordance to results from learning psychology trying to direct the attention of the learners to the most relevant parts first. This will give them the impression of a well-organised system.

2.2. Abstraction layers in e-learning systems

In this section we present a framework for the design of e-learning systems. The framework is based on an *Abstraction Layer Model*, which is illustrated by Figure 1. From top to bottom

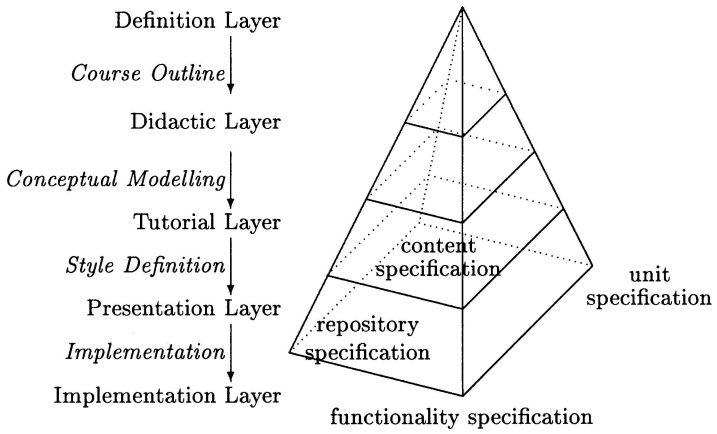

Figure 1. Abstraction layers in web-based e-learning systems.

we identify a definition layer, a didactic layer, a tutorial layer, a presentation layer, and an implementation layer.

The general ideas of this model are as follows. We identify several layers of abstraction:

- The top layer is called the *definition layer*. It is used to describe the system in a general way: What is the purpose? What are the course goals? Who are the expected learners?
- The next lower layer is called the *didactic layer*, which is used to concretise the ideas gathered on the definition layer. This means to get a clearer picture of the different learners and their profiles. The major part of this layer, however, deals with the outline of the course. That is to identify course units and paths through these units.
- The central layer is the *tutorial layer*. Whilst the didactic layer did not pay much attention to technical issues, they now come into play. The various units appearing in the course outline have to be analysed and integrated, so that each unit can be supported by a combination of some data content with some functionality.
- The next lower layer is the *presentation layer* which is devoted to the problem of associating presentation options. Finally, the lowest layer is the *implementation layer*, which addresses all aspects of the physical implementation.

On each layer except the definition layer we identify two dimensions for the description of the system: *focus* and *modus*. The focus dimension distinguishes between local and global components; the modus dimension distinguishes between static and dynamic components.

Global and static components will be addressed by a *repository specification*, which covers all aspects of central data storage and retrieval. Global and dynamic components will be addressed by a *functionality specification*, which covers all operation on the stored data.

Local and static components will addressed by a *content specification*, which covers all aspects of the content of course units. Finally, local and dynamic components will be

addressed by a *unit specification*, which covers the learner activities associated with the learning units.

Each layer is associated with layer specific modelling tasks. The transition from the definition to the didactic layer is associated with developing the course outline(s) and the identification of learner profiles. The transition from the didactic layer to the tutorial layer is associated with conceptual modelling, which addresses database modelling, operations modelling, view modelling, and unit content modelling. This adds more details to the design, as we look at single course units, whereas the composition of the course as such has been dealt with on the didactic layer.

The transition to the presentation layer is associated with the definition of presentation styles. Finally, the transition to the implementation layer is associated with all implementation tasks.

3. Course Modelling

Let us now look at ways to describe the navigation of learners through an e-learning system, for which we may exploit finite, directed graphs for this purpose.

Thus, an *outline graph* is a finite, directed graph $\mathcal{G} = (V, E)$, i.e. V and E are finite sets with $E \subseteq V \times V$. The vertices, i.e. the elements of V, are called *learning units*, and the edges, i.e. the elements of E are *links* between these units.

Take for example a course dealing with e-learning systems. Then we might have learning units such as Introduction, Learner Profiling, Course Outlining, Data Management, Adaptivity, Style Definition, and Implementation. Figure 2 gives a rough picture of the corresponding course outline with links naturally represented by edges.

3.1. Outline graphs

With each learning unit $u \in V$ we associate a view C^u describing the data content of this learning unit. In addition, we may also associate learner types $LT^u_1, \ldots, LT^u_{n_u}$ with the

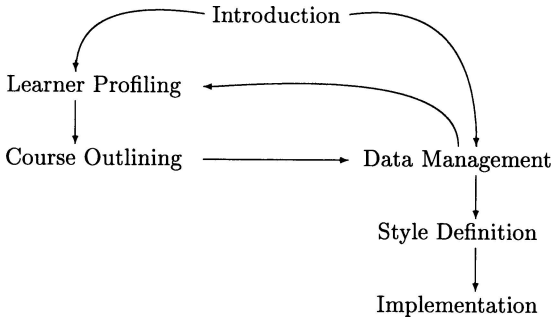

Figure 2. Course outline (sketch).

learning unit. These indicate that the learning unit is suitable for learners of this type only. We will describe learner types in Section 5.

Each link $\ell \in E$ from u_1 to u_2 corresponds to a possible transition from the source learning unit u_1 to the target learning unit u_2. Such a transition should be triggered by an action initiated by the learner. This action can simply be a navigation, but in general we may think of more complicated actions. Therefore, each link is associated with the name of an action that can be issued in that learning unit. In addition, it is also associated with a data type expressing the information communicated from learning unit u_1 to learning unit u_2.

Actions on a learning unit may depend on the successful completion of the learning unit by the learner. According to our view this is part of the action specification, and should be left for further refinement of the outline.

From a more conceptual point of view we model learner interaction with an e-learning system according to two primitives: transition between learning units and using the functionality offered at a given learning unit. This allows for a two-step modelling procedure to be applied. First at a coarse-grained level learning units and navigation links are modelled. Then in a refinement step the actual activities of the the learners are added. This procedure allows for a good separation of concern with respect to personalisation and localisation.

At first the usage of the system by a particular learner type is modelled. This already reduces the complexity of the problem. Then the learner behaviour is dealt with locally, i.e. with respect to a given learning unit, which further reduces complexity. The price for this advantage is the need to integrate the various outlines and the data used within them.

Instead of emphasising the transitions between learning units and the triggering learner activities as sketched in Figure 2, we may want to emphasise the data communication between learning units. As we assume that for any two learning units $u_i, u_j \in V$ there is at most one link from u_i to u_j, the outline graph can be represented by its adjacency matrix $\mathcal{A} = \{a_{i,j}\}_{1 \le i,j \le n}$ with

$$a_{i,j} = \begin{cases} 1 & \text{if there is a link from } u_i \text{ to } u_j \\ 0 & \text{else} \end{cases}$$

Using this representation has the advantage that we get rid of crossing labelled edges in diagrams. However, we use it in a modified form.

First we use a table T with n columns and rows to represent the matrix. We further fill the table's cells $T(i, i)$ for all $i = 1, \ldots, n$ with the name of learning unit u_i and fill the table cell $T(i, j)$ with the labels attached to the edge from u_i to u_j.

The learner types associated with the learning unit u_i will be added at the right side of the table in the form of an additional column. This extended adjacency matrix will be called the *communication matrix* of the course outline.

Figure 3 contains a sketch of the communication matrix for the course outline from Figure 2. We abbreviated the names of the learning units in the obvious way. As we did not yet indicate, which learner activities trigger the navigation links, we only put a • into the corresponding cell of the matrix. Similarly, as we left the learner types open so far, we could only add LT_x into the corresponding cell of the matrix.

Intr	•		•			LT_x
	LP	•				LT_x
		CO	•			LT_x
	•		DM	•		LT_x
				SD	•	LT_x
					Impl	LT_x

Figure 3. Communication matrix (sketch).

3.2. Algebraic modelling of course outlines

Let us take now a closer look at the process algebra `SiteLang` from Thalheim and Düsterhöft (2001). So, let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be a set of learning units, and let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_k\}$ be a set of (atomic) actions. Furthermore, assume a mapping $\sigma : \mathcal{A} \to \mathcal{S}$, i.e. with each action $\alpha \in \mathcal{A}$ we associate a learning unit $\sigma(\alpha)$.

This can be used to define inductively the set of *learning processes* $\mathcal{P} = \mathcal{P}(\mathcal{A}, \mathcal{S})$ determined by $\mathcal{A}$ and $\mathcal{S}$. Furthermore, we can extend σ to a partial mapping $\mathcal{P} \to \mathcal{S}$ as follows:

- Each action $\alpha \in \mathcal{A}$ is also a learning process, i.e. $\alpha \in \mathcal{P}$, and the associated learning unit $\sigma(\alpha)$ is already given.
- 0 and 1 are learning process, for which σ is undefined. 1 is a content-less learning process, while 0 stands for a failed learning process.
- If p_1 and p_2 are learning processes, then also the *sequence* $p_1 \cdot p_2$ is a process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is 1, then $\sigma(p_1 \cdot p_2)$ is also defined and equals s, otherwise it is undefined.
- If p_1 and p_2 are learning processes, then also the *choice* $p_1 + p_2$ is a learning process. Furthermore, if $\sigma(p_1) = \sigma(p_2) = s$ or one of the p_i is 1, then $\sigma(p_1 + p_2)$ is also defined and equals s, otherwise it is undefined.
- If p is a learning process, then also the *iteration* p^* is a learning process with $\sigma(p^*) = \sigma(p)$, if $\sigma(p)$ is defined.
- If p is a learning process and φ is a boolean condition, then the *guarded learning process* $\varphi \cdot p$ and the *post-guarded learning process* $p \cdot \varphi$ are learning processes with $\sigma(\varphi \cdot p) = \sigma(p \cdot \varphi) = \sigma(p)$, if $\sigma(p)$ is defined.

We also use conjunction $\varphi \cdot \psi$, disjunction $\varphi + \psi$ and negation $\bar{\varphi}$ for boolean conditions. Furthermore, let 1 represent 'true' and 0 'false'. Then the set $\mathcal{P}$ of learning processes carries the structure of a Kleene algebra with tests (Kozen, 1997), i.e. the following rules hold:

- $+$ and $\cdot$ are associative, i.e. for all $p, q, r \in \mathcal{P}$ we must have $p + (q + r) = (p + q) + r$ and $p(qr) = (pq)r$;
- $+$ is commutative and idempotent with 0 as neutral element, i.e. for all $p, q \in \mathcal{P}$ we must have $p + q = q + p$, $p + p = p$ and $p + 0 = p$;
- 1 is a neutral element for $\cdot$, i.e. for all $p \in \mathcal{P}$ we must have $p1 = 1p = p$;
- for all $p \in \mathcal{P}$ we have $p0 = 0p = 0$;
- $\cdot$ is distributive over $+$, i.e. for all $p, q, r \in \mathcal{P}$ we must have $p(q + r) = pq + pr$ and $(p + q)r = pr + qr$;
- p^*q ist the least solution x of $q + px \leq x$ and qp^* is the least solution of $q + xp \leq x$, using the partial order $x \leq y \equiv x + y = y$.

We further adopt the convention to write pq for $p \cdot q$, and to assume that $\cdot$ binds stronger than $+$, which allows us to dispense with some parentheses. It is also known that the double use of $\cdot$ for sequence and conjunction and of $+$ for choice and disjunction, respectively, does not cause any problems. Obviously, if we restrict to boolean conditions only, then the laws of Boolean algebras apply to conjunction, disjunction and negation.

Furthermore, the association of learning units with learning processes implies that we also have sorts. As processes associated with different scenes express concurrency, we claim that $p_1 p_2 = p_2 p_1$ holds for all $p_1, p_2 \in \mathcal{P}$ with $\sigma(p_1) \neq \sigma(p_2)$, which leads to a many-sorted Kleene algebra with tests (Schewe and Thalheim, 2005).

Example 1. The following expression may express a course outline with actions α_i and conditions φ_i.

$$\begin{aligned}&\alpha_1((\varphi_0\alpha_2 + \varphi_1\alpha_3(\alpha_5 + 1)\varphi_3 + \varphi_2\alpha_4(\alpha_6 + 1)\varphi_4)^*\varphi_5)(\alpha_7\varphi_6 + \alpha_{13}\varphi_7)\\ &\quad(\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8 + \varphi_7\alpha_8\alpha_8^*\alpha_{14}\alpha_{15}\alpha_{16}^*\alpha_{11}\alpha_{17}(\overline{\varphi_{12}}\alpha_{18}\alpha_{19})^*\varphi_{12}\alpha_{18}\varphi_9)\\ &\quad\alpha_{20}(\varphi_{10} + \varphi_{11})\end{aligned}$$

In addition we may have the following assignment of learning units s_i to the actions

$$\begin{aligned}&\sigma(\alpha_1) = s_1 \quad \sigma(\alpha_2) = s_2 \quad \sigma(\alpha_3) = s_2 \quad \sigma(\alpha_4) = s_3 \quad \sigma(\alpha_5) = s_2\\ &\sigma(\alpha_6) = s_3 \quad \sigma(\alpha_7) = s_4 \quad \sigma(\alpha_8) = s_4 \quad \sigma(\alpha_9) = s_4 \quad \sigma(\alpha_{10}) = s_4\\ &\sigma(\alpha_{11}) = s_4 \quad \sigma(\alpha_{12}) = s_4 \quad \sigma(\alpha_{13}) = s_5 \quad \sigma(\alpha_{14}) = s_5 \quad \sigma(\alpha_{15}) = s_5\\ &\sigma(\alpha_{16}) = s_5 \quad \sigma(\alpha_{17}) = s_5 \quad \sigma(\alpha_{18}) = s_5 \quad \sigma(\alpha_{19}) = s_5 \quad \sigma(\alpha_{20}) = s_6\end{aligned}$$

with the learning units

$$\begin{aligned}&s_1 = \text{Introduction} \quad s_2 = \text{Learner Profiling} \quad s_3 = \text{Course Outlining}\\ &s_4 = \text{Data Management} \quad s_5 = \text{Style Definition} \quad s_6 = \text{Implementation}\end{aligned}$$

4. Linguistic Analysis

Linguistic analysis summarises the activities and techniques that are applied to analyse natural language descriptions of learners' activities. The major source of information used

for developing e-learning systems is a description of curriculae, which may be delivered in the form of texts or verbally.

Linguistic analysis takes such natural language descriptions and uses them for analysing the activities of the learners: Which are these activities? Does the description indicate any sequencing or continuation? What data is needed for or used by the activities? What are the relationships between these data?

As learner activities can be described by verbs, we suggest to analyse the corresponding word fields. We will show that word fields are a valuable source for centering the outline graph around the central learner activities.

4.1. Word fields

According to Hausser (2001) a word is an abstract concept, which in order to become concrete needs a word form carrying the grammatical variants. Word fields are a much more general notion than word forms. Word fields combine different aspects:

- morphology, i.e. the forms of the written or spoken word,
- phonology, i.e. the sound of the spoken word,
- syntax, i.e. the construction of sentences using the word forms,
- semantics, i.e. the meaning(s) of the word, and
- pragmatics, i.e. the usage of the word in written or spoken language.

Our restriction to written language communication makes considering phonology obsolete. Under this premise we understand a *word field* to consist of its intext, its context, its semantics and its pragmatics.

- The *intext* combines morphology and syntax. We are mainly interested in the latter one, especially for verbs. In this case we may ask for the basic syntactical form, various mandatory and optional arguments, variants of the arguments, extensions to the basic syntax, and relationships and constraints between the arguments of the word when it is used in sentences.

  For instance, the verb "*to teach*" uses a mandatory argument "*something*" and an optional argument "*to somebody*". The basic syntactical form "*Someone teaches something to somebody*" expresses an action, specifies the actor and the receiver, and an object used in the action.
- The *context* refers to application areas. According to the context we may identify related words, concepts related to the arguments, categories of actors and actions, and further constraints.

  For instance, in a learning context the verb "*to teach*" may be used in a sentence such as "*The teacher teaches e-commerce to third-year students*". In this case we conclude that the object of the action is subject to an assessment of whether (s) he has learnt and understood the learning object "e-commerce". It also implies that a follow-on action, i.e. the study by the student, is expected.

– The *semantics* refers to the meaning of a word in a particular context. For example, the already used sentence "*The teacher teaches e-commerce to third-year students*" includes the meaning of the teacher delivering and explaining material and the student working through it and reflecting it in a critical way.
– The *pragmatics* refers to how the word is used in the application context.

4.2. Word fields and outline graphs

Using word fields for the design of dialogues has already been suggested in Thalheim and Düsterhöft (2004). We suggest a compositional approach to outline graph design based on generalised phase structure grammars with applicability rules such as immediate dependence and linear precedence rules (Hausser, 2001).

We propose to exploit word fields in an analytic way starting with the syntactic analysis of sentences. For this of course we also need a sophisticated parsing theory (Hausser, 2001) to obtain the syntax trees. Within such a tree we can identify the main verb and the used arguments. The verb can be used as a descriptor for the main activity. The word field of the verb will tell us, which arguments are expected. This will give us hints for detecting the required information and the learners involved.

Finally, the various constraints indicate relationships to other activities, e.g. follow-on activities described by other sentences or relationships to objects used in these other sentences. This gives rise to determine links in the outline graph and the communicated data associated with these links. It may also indicate that the learning unit is still too broad and needs to be refined. For instance, we may exploit the specialisation of verbs or objects that are used as their arguments to the refinement of learning units.

4.3. Metaphors

According to Thalheim and Düsterhöft (2000) metaphors are language expressions used in an uncommon language context. Properly applied, they can simplify the task of communicating complex ideas and result in enthusiastic users.

We already pointed out that generating the best learner support depends on having a model of the learners, i.e. being able to group learners into reasonably constructed learner types. The e-learning system must be designed such that it stimulates questions by the learners that can be answered relatively easily. Metaphors can help in this regard. They can help learners to understand what they can, should, or should not attempt to do next, and allow the learner to master his/her own learning style.

Human communication is largely metaphorical. It is likely that the lack of metaphors in traditional human-computer interaction is responsible for some of the problems of this interaction. Improving human-computer interaction by augmented use of metaphors has the potential to reduce the number of communication barriers as well as their implications.

Recall that computer applications have at least three language levels:

tool language: This is the language in which the learner interface signals the customer the semantics of its functionality.

discourse language: This is the language used in the universe of discourse to identify problems, their solutions and quality criteria of all of them.

metaphor language: This language helps the learners understand the state of affairs in the universe of discourse and what interface functions can be used to achieve their goals.

The generation of metaphors could be supported by logically decomposing the learning space into a small number of domains that appear homogeneous with respect to the offered functionality. This permits relating the domains to learner types and expected learner activities. The generation of metaphors appears then as being connected to finding characterising names for the relationships bewteen leaner types, expected learner activities and domain.

5. Learner Modelling

In this section we focus on the learner model following an inspiration by the whole-person-approach from Martinez (2001). In this approach a comprehensive set of human factors for learner modelling is considered. This leads to conceptual modelling of learners by learner profiles, i.e. value combinations for a list of learner characteristics. A complete list of such characteristics will of course depend on the learning domain. Therefore, we concentrate on selected reasonable characteristics.

5.1. The learner space

We follow Martinez (2001) in considering cognitive, conative, and affective personality aspects of individuals as key regarding the outcomes of their learning processes. These psychological aspects impact the learning outcome. Increasing the learning performance of most learners, if based on adaptation of e-learning systems to learners, thus appears to require learner characteristics being identified that relate to the mentioned personality aspects. We shall distinguish between characteristics of the learning style, the learner as a person, the learning task at hand, and the preferred examination style. The following lists are assumed to be non-exhaustive.

For the learning style we have the following general characteristics:

Guidance: The degree to which the learner prefers being guided or not in the learning process.

Visual Modality: The degree to which the learner prefers a presentation in visual manner. We follow (Rostanin *et al.*, 2002) in using the characteristic modality.

Auditory Modality: The degree to which the learner prefers a presentation in auditory manner.

Textual Code: The degree to which learners with visual learning style prefer having access to text.

Illustrational Code: The degree to which learners with visual learning style prefer having access to images.

Example: The degree to which the learner prefers learning deductively or inductively.

Characteristics that refer to the learners are:

Persistency: The degree to which the learner in general can or cannot mentally cope with new material.
Retentivity: The degree to which the learner in general can or cannot memorize learned material.
Computer literacy: The degree to which the learner has or has not acquired skills in using modern computing infrastructure for a task at hand.
Curiosity: The degree to which the learner in general is or is not fond of learning new material.

Learning task related characteristics are the following:

Prerequisites: The degree to which the learner has or has not learnt the required subjects.
Performance: The degree to which the learner has or has not performed well regarding background subjects, required subjects or subjects in general.
Motivation: The degree to which the learner is or is not interested in efficiently mastering the learning task at hand.
Confidence: The degree to which the learner believes to be capable or incapable of successfully mastering the learning task.

Characteristics that apply to the examination style are:

Kind: The degree to which the learner either prefers his knowledge or his skills or a combination of both being checked.
Control: The degree to which the learner prefers his increment in knowledge or skills being checked.
Evaluation strategy: The degree to which the learner prefers his solution to an examination problem being evaluated right after its submission or whether he/she prefers all problem solutions being evaluated at the same time.
Feedback: The degree to which the learner prefers receiving full explanations of the assessments of his/her problem solutions or whether he/she prefers being notified of the correct results only.

We believe that the chosen grouping of characteristics is obvious and that the characteristics we have defined are obvious and plausible candidates for characteristics in real learning tasks. We further believe that at least extreme scores in scales for the characteristics result in manifest options in designing and presenting learning units. We thus do not discuss these issues here in more detail. Clearly, persistency and retentivity are cognitive; curiosity and motivation are conative; and guidance, evaluation strategy and feedback are affective personality aspects. Thus indeed our characteristics for the key personality

aspects gives at least a hint on how to break them down and make use of these for e-learning systems.

5.2. *Formalising the learner space*

In order to describe learner profiles we identified general and subject-specific characteristics of learners. So formally, we obtain a non-empty set C of *learner characteristics*. In addition, for each of these learner characteristics we obtain a linearly ordered set of values, which from now on will be called the scale of the characteristic. Formally, for each characteristic $c \in C$ we have a *scale* $S(c)$ and a linear order $\leq$ defined on it.

For example, if the characteristics are abstraction, perception and memory, we may use a numerical scale, say the integer interval $[0, \ldots, 5]$, for each of them.

- A low value for abstraction indicates a learner who needs more concrete examples in order to be able to understand the learning material, whereas a higher level indicates a better ability of abstract thinking.
- A low value for perception indicates a need for visualisation, while a higher value indicates that the learner can cope with textual and maybe even formal writing.
- A low value for memory indicates that the learner often needs to review larger parts of the learning material, whereas a higher value indicates that this is not needed or that the learner only needs a brief reminder in a very condensed form.

The *learner space LS* is the cartesian product of the scales, i.e.

$$LS = \prod_{c \in C} S(c).$$

Thus, each element of the learner space is a tuple, and each component of this tuple indicates the value of a certain learner characteristic. That is, the learner space captures our knowledge about the different combinations of learner characteristics.

Figure 4 illustrates a learner space with two characteristics, i.e. $C = \{c_1, c_2\}$, and the scales $S(c_1) = \{v_1, v_2, v_3, v_4\}$ with $v_1 < v_2 < v_3 < v_4$, and $S(c_2) = \{v'_1, v'_2, v'_3\}$ with $v'_1 < v'_2 < v'_3$. The learner space $LS = S(c_1) \times S(c_2)$ contains 12 tuples.

5.3. *Learner types*

Thus, the learner space is an adequate way to represent the different learners. However, we have also seen that there are not so few characteristics of learner types, and for each of them we obtain quite a few different values to be taken into account. This bears the risk of a combinatorical explosion of the learner space.

In order to personalise a system to learners, we would have to parameterise each learning unit and each navigation link by the elements of the learner space or provide functions that map units and links between them to a personalised course outline. This is only feasible, if

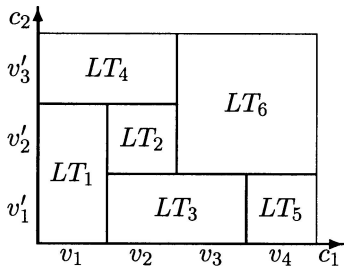

Figure 4. Convex learner types.

either the learner space is small or the personalisation will be the same for many elements of the learner space.

For instance, if we have 10 different characteristics and each comes with only two different values in their scale, then we would already have $2^{10} = 1024$ elements in the learner space. This is far too much.

Therefore, we have to bundle points in the learner space, i.e. instead of attempting to personalise the system to each combination of characteristic values, we would personalise it only to learner types. In general, a *learner type* should correspond to several points in the learner space. We now discuss two different ways to combine elements of the learner space to learner types.

A *convex learner type LT* is a cuboid in *LS*. In other words, if $C = \{c_1, \ldots, c_n\}$, then take intervals $[v_i, v_i'] \subseteq S(c_i)$ for all scales $i = 1, \ldots, n$ and define

$$LT = \prod_{i=1}^{n} [v_i, v_i'] \subseteq LS.$$

Look again at Figure 4. Here we defined six convex learner types:

$$LT_1 = \{v_1\} \times \{v_1', v_2'\} \quad LT_2 = \{v_2\} \times \{v_2'\}$$
$$LT_3 = \{v_2, v_3\} \times \{v_1'\} \quad LT_4 = \{v_1, v_2\} \times \{v_3'\}$$
$$LT_5 = \{v_4\} \times \{v_1'\} \quad LT_6 = \{v_3, v_4\} \times \{v_2', v_3'\}$$

In other words, for learners with value v_1 for characteristic c_1 we do not make a distinction between the values v_1' and v_2' anymore, as this will not have an effect on the system design.

Alternatively, we may apply aggregation functions to the learner space. In this case we assume that the scales are sets of integers, e.g. intervals $S(c_i) = [m_i, M_i]$. An *aggregate function* on the learner space is an integer valued function $f : LS \to \mathbb{N}$.

An *aggregate learner type* is a subset of *LS* of the form

$$LT = \{\ell \in LS \mid m \leq f(\ell) \leq M\}$$

for some integer interval $[m, M]$.

For instance, in Figure 5 we have taken the integer scales $S(c_1) = \{0, 2, 3, 5\}$ and $S(c_2) = \{0, 1, 4\}$. Then the aggregation function is a simple addition, i.e. $f(v, w) = v + w$.

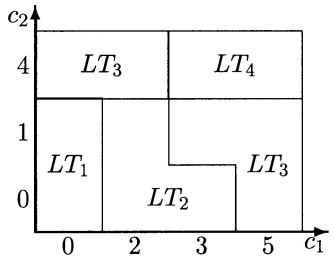

Figure 5. Aggregate learner types.

We then define four aggregate learner types:

$$LT_1 = \{(v, w) \in LS \mid 0 \leq f(v, w) \leq 1\}$$
$$LT_2 = \{(v, w) \in LS \mid 2 \leq f(v, w) \leq 3\}$$
$$LT_3 = \{(v, w) \in LS \mid 4 \leq f(v, w) \leq 6\}$$
$$LT_1 = \{(v, w) \in LS \mid 7 \leq f(v, w) \leq 9\}$$

These learner types are illustrated as well in Figure 5.

5.4. Intentions, rights and obligations

The presence of different learner types indicates different usage of the system. An *obligation* of a learner type specifies what a learner of that type has to do. A *right* specifies what a learner of that type is permitted to do. Both obligations and rights together lead to complex deontic integrity constraints. We use the following logical language $\mathcal{L}$ for this purpose:

- All propositional atoms are also atoms of $\mathcal{L}$.
- If α is an action on learning unit s and r is a learner type associated with s, then $\mathbf{O}\ do(r, \alpha)$ is an atom of $\mathcal{L}$.
- If α is an action on learning unit s and r is a learner type associated with s, then $\mathbf{P}\ do(r, \alpha)$ is an atom of $\mathcal{L}$.
- If α is an action on learning unit s and r is a learner type associated with s, then $\mathbf{F}\ do(r, \alpha)$ is an atom of $\mathcal{L}$.
- For $\varphi, \psi \in \mathcal{L}$ we also have $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \Rightarrow \psi$ and $\varphi \Leftrightarrow \psi$ are also formulae in $\mathcal{L}$.

The interpretation is standard. In particular, $\mathbf{O}\ do(r, \alpha)$ means that a learner of type r is obliged to perform action α, $\mathbf{P}\ do(r, \alpha)$ means that a learner of type r is permitted to perform action α, and $\mathbf{F}\ do(r, \alpha)$ means that a learner of type r is not allowed to perform action α.

Example 2. If action α_8 represents an indispensible part of the course, e.g. a compulsory assignment, each learner will have to take it, once one of the actions α_7 or α_{13} have been executed (condition φ_6 or φ_7). Furthermore, assume that in case of φ_7 the learner is obliged to execute actions α_{16} and α_{17}, i.e. we obtain the deontic constraints

$$\varphi_6 \vee \varphi_7 \Rightarrow \mathbf{O}\ do(\text{learner}, \alpha_8)$$

and

$$\varphi_7 \Rightarrow \mathbf{O}\ do(\text{learner}, \alpha_{16}) \wedge \mathbf{O}\ do(\text{learner}, \alpha_{17})$$

The *intention* of a learner can be expressed by goals, which can be modelled by post-conditions to the learning space, which itself is described by an expression in the algebra discussed in Section 3.

6. Course Personalisation

Let us now discuss some techniques for the personalisation of outline graphs: reduction to a subgraph, and splitting of learning units.

6.1. *Reduction to a subgraph*

According to the definition of outline graphs each learning unit is associated with a set of learner types. As a consequence, given a certain learner type only those learning units that are associated with it, are accessible for a learner of this type. This leads to the definition of the subgraph spanned by a learner type.

Formally, let LT be some learner type, and let $\mathcal{G} = (V, E)$ be an outline graph. Then the set of learning units that are accessible to learners of type LT is

$$V(LT) = \{u \in V \mid u \text{ is associated with } LT\}.$$

The outline graph $\mathcal{G}(LT) = (V(LT), E(LT))$ with

$$E(LT) = E \cap (V(LT) \times V(LT))$$

defines the *subgraph spanned by LT*, i.e. we only consider those links, for which the starting and the target learning unit are in the set of learning units that are accessible to learners of type LT.

Of course $\mathcal{G}(LT)$ is an outline graph, and all its links are associated with the learner type LT. In fact, this subgraph models exactly the part of the outline graph that corresponds to the permitted navigation for learners of type LT.

We may also exploit the algebra from Section 3 to compute a subgraph by equational reasoning. The general approach is to formulate a problem by using equations or conditional equations. Furthermore, we obtain (conditional) equations, which represent application knowledge. This application knowledge arises from events, postconditions and knowledge

about the use of the system for a particular purpose. We then apply all equations to solve the particular problem at hand.

The application knowledge contains at least the following equations:

1. If an action p has a precondition φ, then we obtain the equation $\bar{\varphi}p = 0$.
2. If an action p has a postcondition ψ, we obtain the equation $p = p\psi$.
3. If an action p is triggered by a condition φ, we obtain the equation $\varphi = \varphi p$.
4. In addition we obtain exclusion conditions $\varphi\psi = 0$ and tautologies $\varphi + \psi = 1$.

The problem of personalisation according to the preferences of a particular learner can be formalised as follows. Assume that $p \in \mathcal{P}$ represents the learner space. Then we may formulate the *preferences* of a user by a set Σ of (conditional) equations. Let χ be the conjunction of the conditions in Σ. Then the problem is to find a minimal process $p' \in \mathcal{P}$ such that $\chi \Rightarrow px = p'x$ holds for all $x \in \mathcal{P}$. Preference equations can arise as follows:

1. An equation $p_1 + p_2 = p_1$ expresses an unconditional preference of action (or process) p_1 over p_2.
2. An equation $\varphi(p_1 + p_2) = \varphi p_1$ expresses an conditional preference of action (or process) p_1 over p_2 in case the condition φ is satisfied.
3. Similarly, an equation $p(p_1 + p_2) = pp_1$ expresses another conditional preference of action (or process) p_1 over p_2 after the action (or process) p.
4. An equation $p_1p_2 + p_2p_1 = p_1p_2$ expresses a preference of order.
5. An equation $p^* = pp^*$ expresses that in case of an iteration it will at least be executed once.

Example 3. Let us continue Example 1. Assume that we have to deal with a learner who knows φ_5. This can be expressed by the application knowledge equation $x\varphi_5 = x$ for all $x \in K$. Furthermore, assume three additional exclusion conditions:

$$\varphi_5\varphi_0 = 0 \quad \varphi_5\varphi_1 = 0 \quad \varphi_5\varphi_2 = 0$$

Taking these equations to the first part of the expression in Example 1 we obtain

$$\begin{aligned}
&\alpha_1((\varphi_0\alpha_2 + \varphi_1\alpha_3(\alpha_5 + 1)\varphi_3 + \varphi_2\alpha_4(\alpha_6 + 1)\varphi_4)^*\varphi_5)x \\
&\quad = \alpha_1((\varphi_0\varphi_5\alpha_2 + \varphi_1\varphi_5\alpha_3(\alpha_5 + 1)\varphi_3 + \varphi_2\varphi_5\alpha_4(\alpha_6 + 1)\varphi_4)^*\varphi_5)x \\
&\quad = \alpha_1((0\alpha_2 + 0\alpha_3(\alpha_5 + 1)\varphi_3 + 0\alpha_4(\alpha_6 + 1)\varphi_4)^*\varphi_5)x \\
&\quad = \alpha_1 1\varphi_5 x \\
&\quad = \alpha_1 x
\end{aligned}$$

That is, the whole learning space can be simplified to

$$\begin{aligned}
&\alpha_1(\alpha_7\varphi_6 + \alpha_{13}\varphi_7)(\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8 + \varphi_7\alpha_8\alpha_8^*\alpha_{14}\alpha_{15}\alpha_{16}^*\alpha_{11}\alpha_{17} \\
&\quad \times (\overline{\varphi_{12}}\alpha_{18}\alpha_{19})^*\varphi_{12}\alpha_{18}\varphi_9)\alpha_{20}(\varphi_{10} + \varphi_{11})
\end{aligned}$$

Personalisation also means satifying the intention of a particular learner. Let this intentions be formalised by the postcondition $\psi \in B$. Furthermore, assume that the learning

space is represented by some process expression $p \in \mathcal{P}$. Then the problem is to find a minimal process $p' \in \mathcal{P}$ such that $p\psi = p'\psi$. In order to find such a p' we have to use again the application knowledge.

Example 4. Let us continue Example 1 and look at a user with intention φ_{10}. Then we express application knowledge by the equations $\varphi_{10}\varphi_{11} = 0$, $\varphi_{10}\varphi_9 = 0$ and $\varphi_6\varphi_7 = 0$.

Then we can simplify $p\varphi_{10}$ with the expression p from Example 1 step by step. First we get $(\varphi_{10} + \varphi_{11})\varphi_{10} = \varphi_{10}$, which can then be used for

$$\begin{aligned}
&(\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8 + \varphi_7\alpha_8\alpha_8^*\alpha_{14}\alpha_{15}\alpha_{16}^*\alpha_{11}\alpha_{17}(\overline{\varphi_{12}}\alpha_{18}\alpha_{19})^*\varphi_{12}\alpha_{18}\varphi_9)\varphi_{10}\\
&\quad = \varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8\varphi_{10} + \varphi_7\alpha_8\alpha_8^*\alpha_{14}\alpha_{15}\alpha_{16}^*\alpha_{11}\alpha_{17}\\
&\qquad (\overline{\varphi_{12}}\alpha_{18}\alpha_{19})^*\varphi_{12}\alpha_{18}\varphi_9\varphi_{10}\\
&\quad = \varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8\varphi_{10}
\end{aligned}$$

Then finally we get

$$\begin{aligned}
&(\alpha_7\varphi_6 + \alpha_{13}\varphi_7)\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8\varphi_{10}\\
&\quad = \alpha_7\varphi_6\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8\varphi_{10} + \alpha_{13}\varphi_7\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8\varphi_{10}\\
&\quad = \alpha_7\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8\varphi_{10}
\end{aligned}$$

This means that the story space can be simplified to

$$\begin{aligned}
&\alpha_1((\varphi_0\alpha_2 + \varphi_1\alpha_3(\alpha_5 + 1)\varphi_3 + \varphi_2\alpha_4(\alpha_6 + 1)\varphi_4)^*\varphi_5)\\
&\quad \alpha_7\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8\alpha_{20}\varphi_{10}
\end{aligned}$$

6.2. *Splitting of learning units*

The reduction to a subgraph obviously leads to an outline graph that is completely associated with the given learner type. However, it does not change any of the data content of the involved learning units. This problem is addressed by the splitting of learning units.

Roughly speaking, if we are given a learning unit in an outline graph that is associated with a given learner type, we may replace this unit by a whole outline graph such that all learning units in this new outline graph are associated with the given learner type. Furthermore, the combined data content of the units in this replacement outline graph should be equivalent to the data content of the original learning unit.

Formally, let LT be some learner type, and let $\mathcal{G} = (V, E)$ be an outline graph. Let $u \in V$ be a learning unit associated with LT. Now take another outline graph $\mathcal{G}_u = (V_u, E_u)$ with the following properties:

- $V \cap V_u = \emptyset$, i.e. the two outline graphs have no common learning units;
- all learning units $v \in V_u$ are associated with the learner type LT;
- the join of the data contents C_v for all $v \in V_u$ is equivalent to the data content C_u;
- there is a distinguished starting learning unit $v_0 \in V_u$ such that all learning units $v \in V_u$ can be reached from v_0;

– there is a distinguished final learning unit $v_f \in V_u$, which can be reached from all learning units $v \in V_u$.

The first three properties express our intention to replace u by the graph $\mathcal{G}_u$ without losing data content. The last two properties are of technical nature to ensure that this replacement is formally correct.

So, we obtain a new outline graph $\bar{\mathcal{G}} = (\bar{V}, \bar{E})$ with $\bar{V} = (V - \{u\}) \cup V_u$, i.e. we replace the learning unit u by all learning units in V_u. Furthermore, a link $(u', u) \in E$ must be replaced by a link $(u', v_0) \in \bar{E}$ preserving all data types and actions associated with it. Finally, a link $(u, u') \in E$ has to be replaced by a link $(v_f, u') \in \bar{E}$ preserving all data types and actions associated with it.

However, if we want to preserve the learning unit u for other learner types, we would first have to double it in $\mathcal{G}$ ensuring that one copy is assocated with LT, and the other copy with all other learner types.

The splitting of a learning unit requires a more detailed specification of the content of learning units. We will therefore postpone a more technical discussion of splitting to end of the Section 7.

7. Data Management

The content aspect of e-learning systems concerns the question: Which information should be provided? This is tightly coupled with the problem of designing an adequate database. However, the organisation of data that is presented to the user via the pages in an e-learning system differs significantly from the organisation of data in the database. We conclude that modelling the content has to be addressed on at least two levels: a logical level leading to databases, and a conceptual level leading to the content of pages. Both levels have to be linked together.

As pages correspond to learning units, an abstract description of such a learning unit would be a pair (u, v), where u is a URL and v is a complex value based on the use of some type system. We decide to call such a pair a *learning object*. The term 'object' is used, because pairs constructed out of an abstract identifier and a complex value are called 'objects' in the context of object oriented databases.

Using classification abstraction as usual in data modelling, we obtain *content types*. In general, every data type can become a *content type of a learning unit*. Thus, if U denotes a learning unit, the *learning objects of type* U are pairs (u, v) consisting of a value u of type *URL* and a value v of the content type of U.

A little subtlety comes in here, which makes the definition still a bit more complicated. When a value of type *URL* appears inside the content value v, this may be a URL somewhere outside the web information system that we want to develop. However, in the case where the URL is an internal one, it will be the URL of another learning object, say of type U'.

Therefore, we extend content types in such a way that instead of the type *URL* we may also use *links* $\ell : U'$ with a unique link name ℓ and a name of a learning unit U'.

In order to obtain a learning object we would have to replace the links $\ell : U'$ by the data type *URL* first. However, the link $\ell : U'$ would force us to use only values of type *URL* that are URLs of the learning unit U'.

Learning objects support an individual learner and only provide a section of the data of the system. The question arises how the data can be described globally. In fact, we would need to combine all the content types and set up a *database schema*. Designing such a database schema underlies completely different quality criteria. For instance, for databases we would like to avoid redundancies as much as possible. We would also have to pay much attention to providing fast and concurrent access to the data.

Therefore, we use a separate level defined by *database types*. Thus, we obtain a description of the static components on two levels: the global or database level, and the local or information unit level. On both levels we employ the same type system.

Database types are more or less defined in the same way as the content types.

7.1. Content databases

In order to define underlying databases we could refer to any data model. Due to its convenience we prefer to adopt a variant of the Entity-Relationship model (Thalheim, 2000). According to this model we have database types on various levels $k \geq 0$. Usually, the types of level 0 are called *entity types*, whereas types on higher levels are called *relationship types*.

A *database type of level k* has a name E and consists of

- a set $comp(E) = \{r_1 : E_1, \ldots, r_n : E_n\}$ of components with pairwise different role names r_i and names E_i of database types,
- a set $attr(E) = \{a_1, \ldots, a_m\}$ of attributes, each associated with a data type $type(a_i)$, and
- and a key $id(E) \subseteq comp(E) \cup attr(E)$,

such that the database types $E_i \in \mathcal{S}$ are all on levels lower than k with at least one database type of level exactly $k - 1$. In the following we often write $E = (\{r_1 : E_1, \ldots, r_n : E_n\}, \{a_1, \ldots, a_m\}, id(E))$ to denote a database type. The first component in this triple is the component set $comp(E)$. The second component is the attribute set $attr(E)$. The third component is the key $id(E)$.

We use this notation to associate two data types with each database type E. These data types are called the *associated data type* of E and the *associated key type* of E. These types are defined as follows:

- The *associated data type* of E, denoted as type(E), is

$$(r_1 : \text{key-type}(E_1), \ldots, r_n : \text{key-type}(E_n), \\ a_1 : type(a_1), \ldots, a_m : type(a_m))$$

- The *associated key type* of E, denoted as key-type(E), is defined analogously with the difference that only those r_i and a_j are considered that occur in $id(E)$.

In particular, if we have a database type E of level 0, then $comp(E)$ is the empty set. This implies that that there are no fields labelled by r_i.

Let us now conclude the presentation of the global database by defining database schemata. A database schema is simply a collection of database types. Of course, if E_i is a component of a database type E, and E is defined in the schema, then E_i must also be defined in the schema.

Formally, we can define a database schema as follows:

A *database schema* $\mathcal{S}$ is a set of database types satisfying the following condition: If $E \in \mathcal{S}$ is a database type, then for all components $r_i : E_i \in comp(E)$ we must also have $E_i \in \mathcal{S}$.

We define the semantics of database schemata by the collection of possible *databases* prescribed by the database schema. Thus, let $\mathcal{S}$ be a database schema. For each database type $E \in \mathcal{S}$ we have defined an associated data type type(E) and an associated key type key-type(E). As these two are indeed data types, they define fixed sets of values, which we call the set of *objects of type* E and the set of *keys of type* E, respectively:

$$\text{Obj}(E) = dom(\text{type}(E)) \quad \text{and} \quad \text{Key}(E) = dom(\text{key-type}(E))$$

As the key $id(E)$ in a database type E is a subset of $comp(E) \cup attr(E)$, each object of type E can be projected to a key of type E. Let $O_{[\text{key-type(E)}]} \in \text{Key}(E)$ denote the projection of the object $O \in \text{Obj}(E)$ to the value of its key.

We use the sets of objects and keys for the database types $E \in \mathcal{S}$ to define a *database over* $\mathcal{S}$ as follows:

A *database* db *over* $\mathcal{S}$ assigns to each database type $E \in \mathcal{S}$ a finite set $db(E) \subseteq \text{Obj}(E)$ of objects of type E such that the following conditions are satisfied:

- Key values are unique, i.e. there cannot be two different $O_1, O_2 \in db(E)$ with $O_{1\,[\text{key-type(E)}]} \neq O_{2\,[\text{key-type(E)}]}$.
- Component values exist in the database, i.e. for each $O \in db(E)$ and each $r : E' \in comp(E)$ there must exist some $O' \in db(E')$ such that $r : O'_{[\text{key-type(E')}]}$ is part of O.

7.2. *Learning units*

The core of a learning unit is defined by a view. A *view* V on a database schema $\mathcal{S}$ consists of a view schema $\mathcal{S}_V$ and a defining query q_V, which transforms databases over $\mathcal{S}$ into databases over $\mathcal{S}_V$.

The underlying datamodel itself is not relevant. The defining query may be expressed in any suitable query language, e.g. query algebra, logic or an SQL-variant, provided that the queries are able to create links.

This leads to the definition of *data unit* based on some type system. The type system must provide base types and type constructors, e.g. record, set and list type constructors. Arbitrary *type expressions* are built by nesting these constructors.

A *data unit* has a name M and consists of a content data type $cont(M)$, which is an extended type expression, in which the place of a base type may be occupied by a pair

$\ell : M'$ with a label ℓ and the name M' of a data unit, and a defining query q_M such that $(\{t_M\}, q_M)$ defines a view. Here t_M is the type arising from $cont(M)$ by substitution of *URL* for all pairs $\ell : M'$.

In order to model functionality operations are added to data units. An *operation* on a data unit M consists of an operation signature, i.e., name, input-parameters and output-parameters, a selection type which is a supertype of $cont(M)$, and a body which is defined via operations accessing the underlying database.

In order to allow the information content to be tailored to specific learner needs and presentation restrictions, data units are extended to learning units. The most relevant extension is *cohesion*, which introduces a controlled form of information loss. Formally, we define a partial order $\leq$ on content data types, which extends subtyping in a straightforward way such that references and superclasses are taken into consideration.

If $cont(M)$ is the content data type of a data unit M, then let $sup(cont(M))$ denote the set of all larger content expressions, i.e. all expressions exp with $cont(M) \leq exp$.

A total pre-order $\preceq_M$ on $sup(cont(M))$ extending the order $\leq$ on content expressions is called an *cohesion pre-order*. Clearly, $cont(M)$ is minimal with respect to $\preceq_M$.

Small elements in $sup(cont(M))$ with respect to $\preceq_M$ define information to be kept together, if possible. An alternative to cohesion preorders is to use *proximity values*, but we will not consider them here.

A *learning unit* is a data unit M extended by operations and a cohesion pre-order $\preceq_M$. There are other extensions beyond cohesion, but these are not relevant for context modelling.

If the defining query is evaluated for a learning unit M, we obtain a complex value v of type t_M. Together with a generated URL u this defines a learning object of type M.

Cohesion enables a controlled form of information decomposition, which can be exploited for the splitting of learning units as indicated in the previous Section 6. If we want to decompose a learning unit or if we are forced to decompose according to user requirements or technical restrictions, then we may choose a minimal elements $t_1 \in sup(cont(M))$ with respect to $\preceq_M$ such that it satifies the representation requirements. Note that if we only provide a preorder, not an order, then there may be more than one such t_1.

Taking just t_1 instead of $cont(M)$ means that some information is lost, but this only refers to the first data transfer. When transferring t_1, we must include a link to a possible successor containing detailed information. In order to determine such a successor we can continue looking at all content data types $t' \in sup(cont(M))$ with $t_1 \not\preceq_M t'$. These are just those containing the complimentary information that was lost. Again we can choose a least type t_2 among these t' with respect to $\preceq_M$ that requires not more than the available capacity. t_2 would be the desired successor.

Proceeding this way the whole communication is broken down into a sequence of suitable units $t_1, t_2, \ldots, t_n$ that together contain the information provided by the learning unit. Of course, the cohesion pre-order suggests that the relevance of the information decreases, while progressing with this sequence. The learner may decide at any time that the level of detail provided by the sequence $t_1, \ldots, t_i$ is already sufficient for his/her needs.

8. Conclusion

In this article we presented a conceptual view of web-based e-learning systems. Starting from a general abstraction layer model we addressed the modelling of courses, the modelling of learners, the personalisation of courses, and the management of data in e-learning systems. Courses are modelled by outline graphs, which are further refined by some form of process algebra. This process algebra actually gives rise to a many-sorted Kleene algebra with tests, which can be used to fomally reason about the whole learning space.

The linguistic analysis of word fields refering to an application domain helps to set up these course outlines. Learners are modelled by classifying value combinations for their characteristic properties. Each learner type gives rise to intentions as well as rights and obligations in using a learning system. Intentions can be formalised as postconditions, while rights and obligations lead to deontic constraints. The intentions can be used for the personalisation of the learning system to a learner type. We showed that we can use equational reasoning in the Kleene algebra to reduce the learning space to a subgraph that represents the learning needs of a particular learner type.

We approached the management of data in an e-learning system on two different levels dealing with the content of individual learning units and the integrated content of the whole system, respectively. This leads to supporting databases and views defined on them. Furthermore, we demonstrated that the adaptivity feature of the learning units can be used to set up a simple algorithmic approach to personalisation based on the splitting of learning units. Our ideas have been partly realised in the DaMiT system (Binemann-Zdanowicz *et al.*, 2003a; Jantke *et al.*, 2003, 2004; Rostanin *et al.*, 2002), a system that addresses learning in the area of data mining.

We believe that the integrated conceptual nature of our approach and its coupling with rigorous mathematical foundations will help to improve e-learning system in a way that learner-driven learning on demand will be better supported. However, we do not claim that our methodology is a panacea for all problems related to e-learning. We acknowledge that not all of the educational discussion in the field has entered our work nor the work of others, as the technological challenges arising from these discussions have not yet found adequate answers. In other words, we will continue our research in order to further improve our approach.

References

Atzeni, P., Gupta, A., and Sarawagi, S. (1998) Design and maintenance of data-intensive web-sites. In *Proceeding EDBT'98*, volume 1377 of *LNCS*, Springer-Verlag, Berlin, pp. 436–450.

Bergstedt, S., Wiegreffe, S., Wittmann, J., and Möller, D. (2003) Content management systems and e-learning systems: A symbiosis? In *Proceedings ICALT 2003*, IEEE Computer Society, pp. 155–159.

Binemann-Zdanowicz, A., Kaschek, R., Schewe, K.-D., and Thalheim, B. (2004) Context-aware web information systems. In *Conceptual Modelling 2004—First Asia-Pacific Conference on Conceptual Modelling*, S. Hartmann and J. Roddick (eds.), volume 31 of *CRPIT*, Dunedin, New Zealand. Australian Computer Society, pp. 37–48.

Binemann-Zdanowicz, A., Schewe, K.-D., and Thalheim, B. (2004) Adaptation to learning styles. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies – ICALT 2004*, C.-K.

Kinshuk, Looi, E. Sutinen, D. G. Sampson, I. Aedo, L. Uden, and E. Kähkönen (eds.), IEEE Computer Society, pp. 121–125.

Binemann-Zdanowicz, A., Schulz-Brünken, B., Thalheim, B., and Tschiedel, B. (2003a) Flexible e-payment based on content and profile in the e-learning system DaMiT. In *Proceedings of E-Learn 2003*, Phoenix (Arizona), USA. Association for the Advancement of Computing in Education.

Binemann-Zdanowicz, A., Thalheim, B., and Tschiedel, B. (2003b) Logistics for learning objects. In *Proceedings of eTrain 2003*.

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M. (2003) *Designing Data-Intensive Web Applications*. Morgan Kaufmann, San Francisco.

Conallen, J. (2003). *Building Web Applications with UML*. Addison-Wesley, Boston.

Feyer, T., Kao, O., Schewe, K.-D., and Thalheim, B. (2000) Design of data-intensive web-based information services. In *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE 2000)*, Q. Li, Z. M. Ozsuyoglu, R. Wagner, Y. Kambayashi, and Y. Zhang (eds.), IEEE Computer Society, pp. 462–467.

Hausser, R. (2001) *Foundations of Computational Linguistics*. Springer-Verlag, Berlin, Germany.

Hübscher, R. (2001) What's in a prerequisite? In *Proceedings ICALT 2001*, T. Okamoto, R. Hartley, Kinshuk, and J. P. Klus (eds.), IEEE Computer Society, pp. 365–368.

Jantke, K.-P., Lange, S., Thalheim, B., Tschiedel, B., Grieser, G., and Grigoriev, P. (2004) Learning by doing and learning when doing: Dovetailing e-learning and decision support with a data mining tutor. In *Proceedings of the 6th International Conference on Enterprise Information Systems*, Porto, Portugal.

Jantke, K.-P., Memmel, M., Rostanin, O., Thalheim, B., and Tschiedel, B. (2003) Decision support by learning-on-demand. In *Proceedings DSE 2003, CAISE Workshops*, pp. 317–328.

Kaschek, R., Matthews, C., Schewe, K.-D., and Wallace, C. (2003) Analyzing web information systems with the Abstraction Layer Model and SiteLang. In *Proceedings of the Australasian Conference on Information Systems (ACIS 2003)*.

Kaschek, R., Schewe, K.-D., Thalheim, B., Kuss, T., and Tschiedel, B. (2004a) Learner typing for electronic learning systems. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies – ICALT 2004*, C.-K. Kinshuk, Looi, E. Sutinen, D. G. Sampson, I. Aedo, L. Uden, and E. Kähkönen (eds.), IEEE Computer Society, pp. 375–379.

Kaschek, R., Schewe, K.-D., Wallace, C., and Matthews, C. (2004b) Story boarding for web information systems. In *Web Information Systems*, D. Taniar and W. Rahayu (eds.), IDEA Group.

Kerres, M. (2001) *Multimediale und telemediale Lernumgebungen: Konzeption und Entwicklung*. Oldenbourg-Verlag.

Kozen, D. (1997) Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, **19**(3), 427–443.

Martinez, M. (2001) Key design considerations for personalized learning on the web. *Educational Technology and Society*, **4**(1).

Merrill, M. D. (1983) Component display theory. In *Instructional-Design Theories and Models: An Overview of Their Current Status*, C. M. Reigeluth, (ed.), Erlbaum, Hillsdale, NJ, pp. 279–333.

Mohan, P. and Brooks, C. (2003) Learning objects on the semantic web. In *Proceedings ICALT 2003*, IEEE Computer Society, pp. 195–199.

ONTO-LOGGING Consortium (2002) User modelling issues in the context of knowledge management. `http://www.ontologging.com/deliverables.htm`.

Qu, C. and Nejdl, W. (2003) Searching SCORM metadata in a RDF-based e-learning P2P network using Xquery and Query-by-Example. In *Proceedings ICALT 2003*, IEEE Computer Society, pp. 81–85.

Ravenscroft, A. and Matheson, M. P. (2001). Carpe diem: Models and methodologies for designing engaging and interactive e-learning discourse. In *Proceedings ICALT 2001*, T. Okamoto, R. Hartley, Kinshuk, and J. P. Klus (eds.), IEEE Computer Society, pp. 74–77.

Rostanin, O., Schewe, K.-D., Thalheim, B., and Tretiakov, A. (2004) Managing the data in electronic learning systems. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies – ICALT 2004*, C.-K. Kinshuk, Looi, E. Sutinen, D. G. Sampson, I. Aedo, L. Uden, and E. Kähkönen (eds.), IEEE Computer Society, pp. 395–399.

Rostanin, O., Tschiedel, B., and Thalheim, B. (2002) Szenario-basiertes e-Learning für adaptive Inhaltspräsentation. In *Proceedings LIT 2002*, K.-P. Jantke, W. Wittig, and J. Herrmann (eds.), Infix Publishers, pp. 330–338.

Schewe, K.-D., Kaschek, R., Matthews, C., and Wallace, C. (2002) Modelling web-based banking systems: Story boarding and user profiling. In *Proceedings of eCoMo 2002*, H. C. Mayr and W.-J. Van den Heuvel (eds.), Springer LNCS, Springer-Verlag, Berlin.

Schewe, K.-D. and Thalheim, B. (2001) Modeling interaction and media objects. In *Proceedings of 5th Int. Conf. on Applications of Natural Language to Information Systems (NLDB 2000)*, E. Métais (ed.), Springer Verlag, Berlin. LNCS 1959, pp. 313–324.

Schewe, K.-D. and Thalheim, B. (2005) Conceptual modelling of web information systems. *Data and Knowledge Engineering*. to appear.

Sessink, O., Beeftink, R., Tramper, J., and Hartog, R. (2003) Author-defined storage in the next generation learning management systems. In *Proceedings ICALT 2003*, IEEE Computer Society, pp. 57–61.

Thalheim, B. (2000) *Entity-Relationship Modeling: Foundations of Database Technology*. Springer-Verlag.

Thalheim, B., Binemann-Zdanowicz, A., and Tschiedel, B. (2003) Content modeling for e-learning services. In *Proceedings of the 7th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2003)*.

Thalheim, B. and Düsterhöft, A. (2000) The use of metaphorical structures for internet sites. *Data & Knowledge Engineering*, **35**, 161–180.

Thalheim, B. and Düsterhöft, A. (2001) SiteLang: Conceptual modeling of internet sites. In *Conceptual Modeling—ER 2001*, volume 2224 of *LNCS*, H. S. Kunii, S. Jajodia, and A. Sølvberg (eds.), Springer-Verlag, Berlin, pp. 179–192.

Thalheim, B. and Düsterhöft, A. (2004) Linguistic based search facilities in snowflake-like database schemes. *Data & Knowledge Engineering*, **48**, 177–198.

Van Duyne, D. K., Landay, J. A., and Hong, J. I. (2002) *The Design of Sites*. Addison-Wesley, Boston.