Modular Development of Multimedia Courseware

Khaldoun Ateyeh, Jutta A. Mülle, Peter C. Lockemann
Department of Computer Science
University of Karlsruhe, Germany
ateyeh@ira.uka.de

Abstract

The use of multimedia in courseware and web-based learning is a current topic. This is influenced by the wide availability and the permanently improving technical possibilities. The employment of multimedia technology in education makes it possible to illustrate and to grasp complex processes and coherences. However, the development of multimedia content is still a very costly and tedious task. In order to reduce the costs and efforts needed to build multimedia contents, it is desirable to organize the content in a modular way so that it can be (re)used and created cooperatively. In this paper, we present a development process for multimedia contents that supports a modular cooperative design of multimedia courseware. We argue that applying modularity to courseware design not only reduces the costs but also allows the development of a high-quality reusable and configurable courseware. Configurable courseware can be adapted to meet the needs and characteristics of different lecturers, students and contexts.

Keywords

Reusable courseware, courseware development, hypermedia

1. Introduction and motivation

The development of multimedia courseware is technically difficult, conceptually iterative and thus altogether a very costly process. Important characteristics that distinguish the development process of multimedia systems from the development process of other software systems are: the interdisciplinarity of the developers (computer scientist, psychologist, graphic designer, media specialist, domain expert), the high requirements on creativity, synthetic design, the consideration of psychological and ergonomical aspects, as well as engineering aspects during the development process [1].

Much of the ongoing work pays little attention to the engineering aspects. Consequently, current courseware tends towards monolithic structures that make it very difficult to extend, maintain, update and reuse the learning content. A first remedy lies in the reuse of multimedia elements, such as animation, simulation, video-clips, pictures, audios, etc.. This kind of content reusability is currently the concern of several research projects [3,4].

This is clearly not sufficient in a scenario of cooperative courseware development shared among a number of institutions of higher learning. Particular to this kind of environment is that the developed multimedia courseware must allow its (re)use by different authors and lecturers in different contexts and for different target groups. Our aim is to build a learning module repository that can be combined into different courses for different students of different universities.

Besides this macro level, where modules[1] are combined to courses, a micro level is also conceivable. At the micro level, modules themselves are configurable; authors are able to blank out parts of a module or to add new ones, and authors are also able to (re)define structural relationships between module parts. Only with such configurable modules it is conceivable that modules can be used for different didactical and pedagogical concepts, for different target groups and in different contexts.

Consequently, we need a module concept that has a strong connection to the learning context. Highly desirable is a module concept that encapsulates a semantic meaning and is strongly related to the learning subject, but still flexible enough so that it can be adapted to different needs and different contexts.

This vision of learning modules bears a strong similarity to software modules. The demands are higher, though, because learning modules show some particularities. So, to make this vision a reality we have to find specific ways and concepts for the development of this kind of modules. This is the subject of this paper. We introduce a concept of learning modules such that these can be adapted to the goals, tasks, interests and other characteristics of individual users, and support reusability in a many-fold manner.

[1] We will use module in the following as a synonym to learning-module

In Section 2 we discuss the requirements for the concept. Section 3 gives an overview of related work. In Section 4 we outline the module concept as the basis of the development process. Section 5 discusses the development process in detail. Finally, we give a summary in Section 6. To illustrate the concept, we use an example scenario, where a group of two authors who come from two different universities develop in cooperation the course unit "Entity-Relationship Modelling" (ER-Modelling) from the subject area "Database Systems".

2. Requirements for the development of multimedia courseware

Below we discuss requirements on courseware to the extent that our development process will have to support them.

Reusability: Reusability is the main focus of our approach. Our goal is not only to support reusability on the level of atomic multimedia elements (animation, simulation, audio, text, etc.) but also on a more semantical level, i.e., the level of learning units.

Support for face-to-face and distance learning: It should be possible to use the developed learning content in both face-to-face and in distance education. This requires the ability to construct different views on the learning materials.

Adaptability:

Adaptability to University types: It should be possible to use the developed learning modules for various target groups, e.g., students of different types of universities. These target groups have different characteristics (background, goals, etc.) that should be considered during the design of the modules.

Adaptability to students: The learning material should satisfy different characteristics of the students such as a student's profile, learning method and background. To meet these requirements, it should be possible to construct different views on the learning material so that students can switch between these.

Adaptability to authors: Different authors have different needs. They use different didactical and pedagogical methods suitable for their target group. It should be possible for authors to derive their own view on the developed materials and to apply their own didactic to it.

Cooperation support: Cooperative development of learning content reduces the enormous efforts and costs for participants.

Open standards: Open standards such as IMS[4] and XML[5] are important so that the developed contents can be easily shared and exchanged between content development and content management systems.

Automation: It should be possible to automate as many steps of the development process as possible. This will reduce the development costs.

Suppose for our example scenario that we have two authors from two different universities where one emphasises theory and the other takes a more pragmatic approach. Thus we have two target groups with different characteristics such as knowledge background and learning goals. While the two authors plan to develop the learning materials cooperatively and, hence, need similar support, the developed material must clearly be adaptable to their individual needs. Moreover, it is quite natural to require that the learning materials should be used both for face-to-face education during lectures and for distance learning.

3. Related work

Recently, a spate of so-called learning systems has appeared on the market. Examples for these products are: IBM LearningSpace [6], Oracle Online Learning Application [7], WebCT [8], Hyperwave Training Space (GENTLE) [9, 10]. Learning systems offer a complete platform for distributed learning, they provide an integrated environment for both students and trainers. They contain management and administration tools for both learning materials and users. These systems also include components for communication and collaboration such as chat, mail, tele-conferencing, news groups, white board, etc.. However, they offer little or no support for the task of authoring the learning materials. Rather, they only contain a simple authoring tool such as a text editor, and leave the other tasks to external authoring tools.

Other recent approaches deal with the development of hypermedia applications [11,12,13]. Their models, such as RMM[12] and OOHDM [13], mostly deal with presentation aspects such as the design of navigation structures among hypermedia objects, but pay little attention to the organisation of the content.

Courseware reuse has been an aspect of the ARIADNE project [3,14]. However, it still limits reuse to basic multimedia elements (animation, text, audio, etc.), i.e., more to form than substance. The HyperSkript project [15] also has an element of reuse by providing a cooperative environment for developing and using distributed computer-based lecture material.

The emphasis of all these approaches on presentation would not be that bad if they pursued a strategy of separation between content and presentation. So far, this has only been a tradition in the software design pattern of Model View Controller (MVC) as reflected in, e.g., SmallTalk [16], Java Swing [17], or in SGML and XML [18], but has not been an issue in learning systems. Our hypothesis is that the requirements of section 2 can best be met by a clear separation between content and

presentation. This paper concentrates on the aspect of content and, hence, deals with modelling hypermedia objects themselves. It sets its priorities on reusability and cooperative design. In particular, our approach supports courseware adaptability so that it meets the characteristics of both authors and learner as well as the different learning forms.

4. Basic approach

The main challenge in this work is that a courseware is on the one hand a self-contained learning unit, and on the other hand needs to be flexible enough to be easily adapted to different courses and contexts. It requires a compromise between so-called "one-size-fits-all" (difficult to change), ready-to-use courseware that can only be used in a single context and just by its author, and configurable courseware that can be adapted for use by different educators and thus in different contexts.

Naturally, course developers tend to the first case because their own needs are foremost on their minds, and they usually are under pressure. That is, development follows precise requirements from one author, for one target group and for one context. All information such as content, appearance, didactic is mixed together in the courseware so that it is very difficult to change one of them independently of the others. It is unlikely that such a courseware can be used by other authors, because authors usually have different needs and want to be able to adapt the courseware -often on short notice- to meet their needs and interests, such as didactic and pedagogical methods, context and target group. On the other hand, it is also inconceivable that the courseware will be used even by its author once he/she has a different target group with different characteristics in mind.

The first step to gain in adaptability is to break down the learning material into learning units, that are thematically self-contained, and for which we may give a motivation and a solution for a specific problem, and which should not be broken down any further if a coherent semantic is to be maintained. Learning units are realised in several steps; each result refered to as a module [21]. If properly done, the modules can be developed, (re)used and maintained independently.

In the next step lecturers can then construct their courses just by choosing the appropriate modules. Depending on the context, different combinations of modules can be chosen to meet the specific characteristics of the context.

This step ensures adaptability at the macro level (see section 1), which means that authors can adapt the course by changing the combination of modules, by choosing another navigation-structure, or by replacing a module with a new one. However the approach offers no support for adaptability on the micro level. Modules are not configurable and have to be used as they are. To meet new requirements new versions of the modules have to be developed, resulting in a proliferation of modules.

To overcome these limitations we introduce a modularisation concept that is tailored to module adaptability. The concept relies on different abstraction levels that allow the construction of different views on a learning unit (see figure 1: the dark-shaped boxes).

We also introduce the concept of module type. This allows us to categorise modules into classes, with the benefit of sharing concepts such as style, didactic etc. among all instances of a module type.

5. Levels of modularisation

Depending on the size of the learning subject, it is represented by one or more course units. Course units are divided into learning units. Participating authors to the development of the courseware identify all the learning material needed for every learning unit. This material is collected into a so-called integration module. Integration modules are content-based and are almost free of didactic consideration, which means that modules on this level contain no information about the learning context, or about the position of the content, or about the appearance of the content. Hence on this level, the learning material still is entirely content-oriented, with little concern for the target groups and lecturers. Beside the total of the learning materials needed by the different users, integration modules contain meta information to describe the semantical meaning of the content.

On the next level, the so-called structural modules emerge via some sort of view construction from a given integration module. View construction has many elements known from database systems: selection of subsections, imposing a structure on the selected parts, augmentation by specialised components. Structural modules are structured particularly with regard to the way they are to be used and to other aspects like learning situation, target group and author’s needs. During the view construction, the author has the possibility to choose those ingredients from the modules that appear suitable for his/her view. He/she can also restructure a module so that it meets his/her didactical and pedagogical methods. Finally, authors can add new ingredients (components) specific to their view.

At the lowest level, the content is augmented by the visual appearance of the modules. By the design of the appearance, consideration has to be given to both psychological and didactical as well as to pedagogical aspects. The result are the so-called presentation modules.

Representation modules are what really used by teachers and students. Given these, courses can be

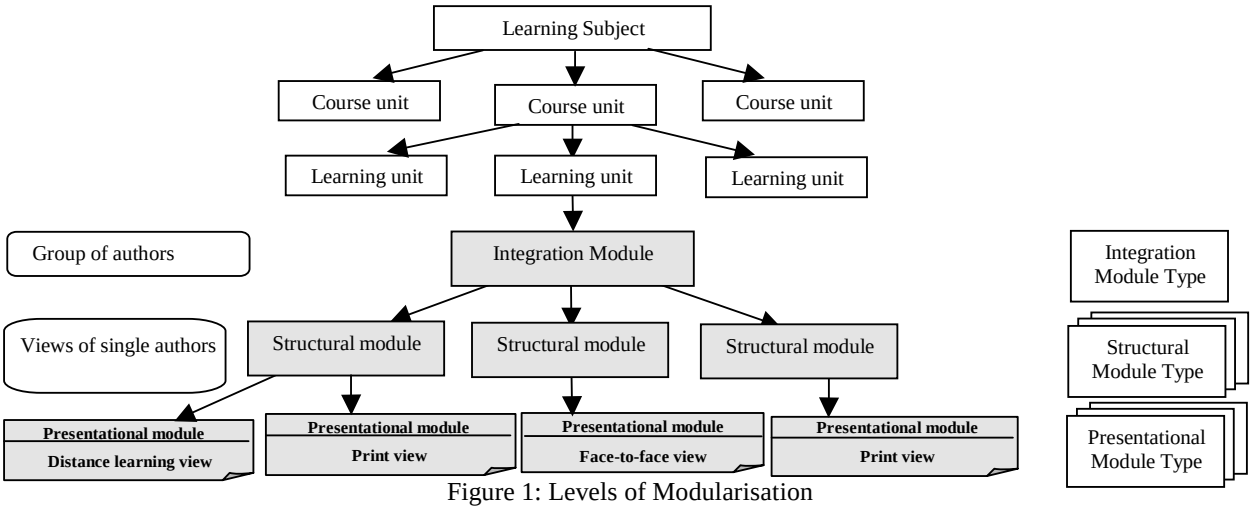

Figure 1: Levels of Modularisation

constructed from the presentation modules. This is done by selecting the needed modules, given proper attention to the logical interdependencies between them. The selected modules are connected using a navigation structure, such as a guided tour.

At all levels of modularisation we distinguish between module type and module instance. Similar to the concepts of class and instance or object in the object-oriented methodology. A module type can be considered a template that imposes a certain pattern on modules and in this way is an additional means for easing reusability. Depending on the level, a module type gives guidance to the collection, organisation and presentation of contents. Module types allow the separation of content from authors specific concepts such as style and didactic that normally makes content reuse a very difficult task. In a first design phase, the module types for the subject area of the learning material are designed. For example a module type could have attributes such as title, description and motivation; a learning module "instance" of this type assigns to the attributes a concrete content. An instance does not need to provide an implementation for every attribute defined in its type; e.g., it is conceivable that one learning module includes a motivation and another does not.

6. The development process

There are clearly two major steps to the development process: module design and course design. The former is critical to the overall issues of adaptability and re-use, the later is critical to the educational success. Thus even though the former takes precedence over the later, the later must feedback on the former (see figure 2).

Module design follows the hierarchy in figure 1. The abstraction layers of figure 1 relate to the development phases of the learning modules. They are mapped, respectively, to content-based design, structural design, and presentational design as the development phases.

The development of the learning modules comprises an evaluation step and, related to the design phases, the corresponding implementation steps. Clearly of course, the development process must be preceded by a requirements analysis phase.

6.1. Requirement analysis

This phase serves to analyse and limit the scope of the subject area. Representatives of all participating institutions cooperate[2].

The goals of this phase are:

- Identification of target groups.
- Identification of learning goals, i.e., restriction of the subject area.
- Identification of the relevant course material.

In our example scenario the target groups are mainly the students of the two universities. Let us assume that one of the two universities is more theory-oriented and the other one more engineering-oriented. We also assume that the course content already exists, that is well-understood course material is to be put into a multimedia form.

6.2. Design

During the development process a learning unit is realised in different stages (see figure 3).

Conceptually, a learning unit spawns different module types as the design moves along the phases. In turn the module types are instantiated to modules.

[2] This does not mean that the developed modules cannot be used by others as well.

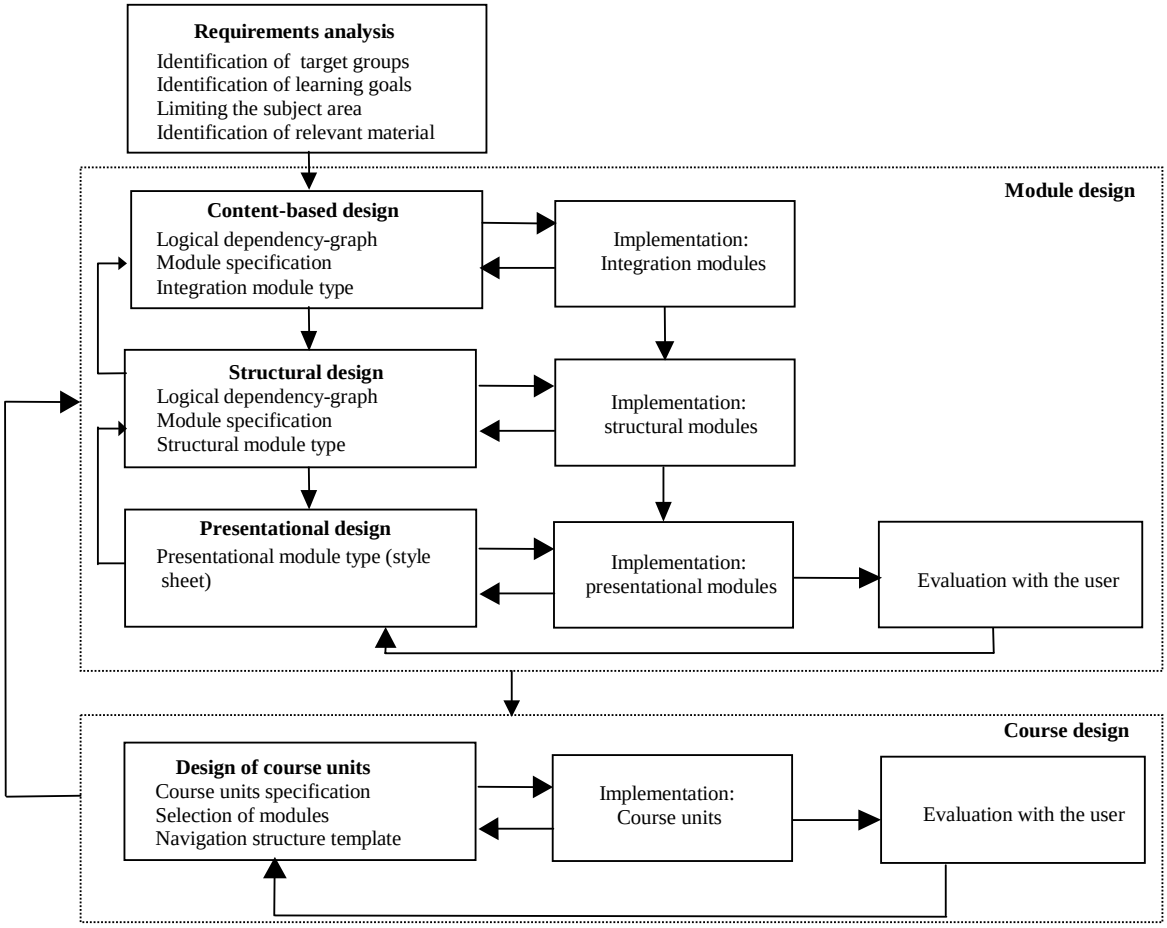

Figure 2: The Development Process

The design process is divided into the following phases that can be alternatively interleaved with the implementation phases:

- Content-based design
- Structural design
- Presentational design

In the following, we discuss the different design phases (see figure 2) in detail.

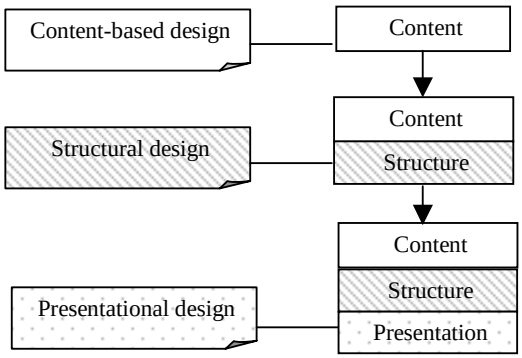

Figure 3: Transition Phases of a Learning Module

6.2.1. Content-based design

The first phase concentrates on identifying the specific content for the course unit. The content is represented by one or more integration module. Each module is divided into components. Integration modules and their components are content based. They indicate what the information is (or represent) in an abstract sense, and avoid implying what its ultimate appearance will be [19]. In our example scenario, an example for an integration module is: "Relationships and Relationship Types" with the components definition, notation (see figure 4). This is done by identifying semantic learning units in the learning materials and dividing them into content-based components. The identified modules are then related to each other, so that they form a so-called logical dependency-graph. The logical dependency-graph shows logical dependencies between the modules, such as module A is requested for the understanding of module B. The logical dependency-graph could be very useful in a later stage for combining modules into courses or course units. Since the modules at this stage are still didactic-free, they do not contain any

information about presentational aspects (such as appearance of the content), or about the environment where the modules are to be used.

The following summarises the results of this phase:

Logical dependency-graph: The logical dependency-graph describes the partition of the course unit in modules and how these modules are logically depending on each others.

Integration modules specification: The specification of an integration module consists of two parts:

- Module content: A list of components that build that module.
- Description information: Information for management purposes (see next "integration module type")

Integration module type: The integration module type defines a container-type. It describes all components that an instance of the integration module type may contain. The integration module type consists of two parts:

Description part: This part models management information such as module name, subject, preconditions, goals etc.. This should make use of meta-standards such as IMS.

Content part: This part models the semantic units of the integration module type. It contains a list of references to the module components definitions. At this stage we would like to point out that it is important to define a standard language for the education community. The standard should provide definition for content-based educational components such as algorithm, definition, summary, introduction, keywords etc.. This will enhance learning content reusability and exchange among content management systems and authoring tools.

Relating to our example scenario "ER-Modelling", the logical dependency-graph for ER-Modelling is shown in figure 4. Figure 4 shows also the ingredients of each module. A possible integration module type for the subject E-R Modelling is shown in figure 5. Using XML-notation we have found XML to offer a good framework for the implementation of the concepts discussed in this paper.

6.2.2. Structural design

During this stage, integration modules are structured according to their use. The structural relationships between the components of the integration modules are defined. This is done by adding structural components that define relations between the components of the integration modules. Important aspects thereby are:

Learning type: In which learning type (face-to-face, distance education, self-education) should the modules be used? For example, modules that are used in a lecture are structured differently from modules that are used in a self-education course.

Target groups: For which target group should the modules be used? Different target groups have different characteristics that have to be taken into consideration by structuring the modules. For example, depending on the target group it may be necessary when possible to begin every learning module with a motivation and to follow every sub-topic with an example. By another more experienced target group examples could be omitted.

Authors' needs and interests: How should the learning contents be didactically prepared? For authors it should be possible to adapt the modules.

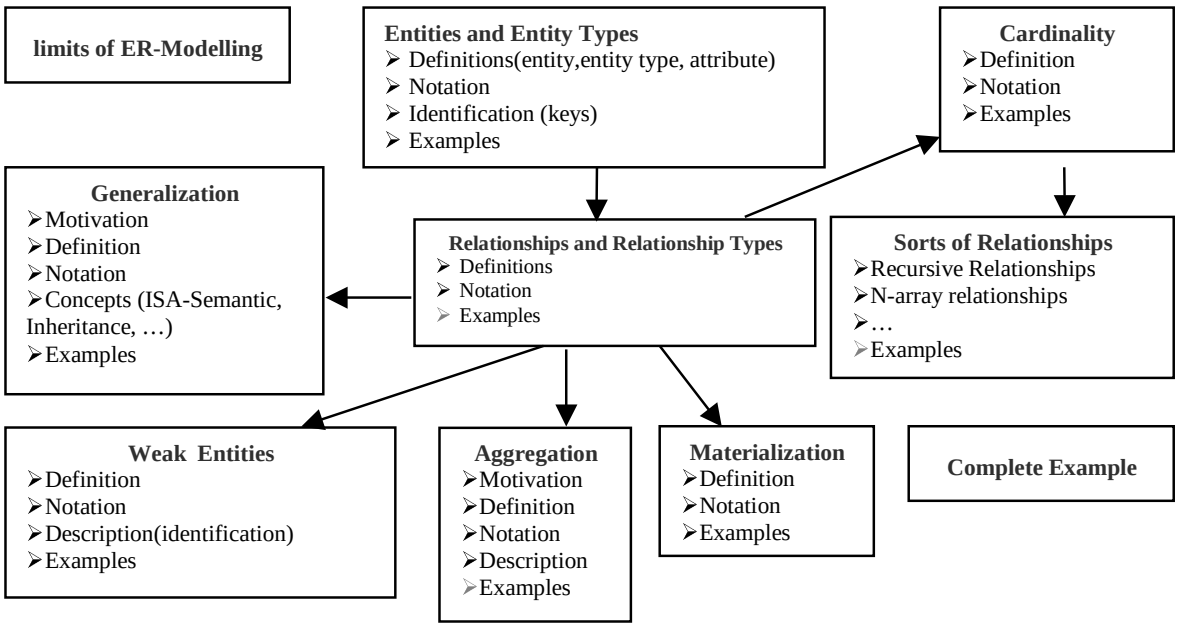

Figure 4: logical dependency-graph „ER-Modelling"

To consider the above-mentioned aspects, views on the integration modules have to be constructed, to meet specific features of individual authors, students and learning types. This is done through the configuration of the integration modules. The resulting views may be classified according to the three aspects: learning type, target groups and author needs. For example, a view could has the characteristics learning type "face-to-face", target-group "university-students" and author "A". The configuration process to construct a view is as follows:

- Identify a view
- Reconstruct the logical dependency-graph by choosing the needed module combination and in some cases adding new modules.
- Select from every integration module instance the components that meet the characteristics of the view.
- If needed, add new components that are not included in the integration module.
- design a structural module type for this view. The designed type complements the integration module type by structural information that defines the relationships between the content-based components so that it meets the requirements of the chosen view. This step could be enhanced by making a structural type repository available. Authors then could simply choose a suitable structural type for their view and don't have to build their own types from scratch. This will enhance reusability and will simplify the development process.

The results of this phase are:

Logical dependency-graph: The Logical dependency-graph in this stage includes only those modules needed for the view.

Module specification: For every identified structural module, a list of the components contained in the module is created.

Structural module type: Similar to the integration module type the structural module type defines a container type. It describes all components an instance of the structural module type may contain. The structural module type also defines the structural relations between its components.

Back to our example scenario, let us assume that both authors are interested to develop material for his/her lecture "face-to-face education", and that both of them decide to build a second view to be used for distance-learning. Accordingly, each of them should perform the above steps for his/her own view (lecture), and, cooperatively, they would build the distance-learning view.

6.2.3. Presentational design

This phase deals with the module appearance to educator and students. It defines precisely how the modules should present themselves and how users interact with them. For each view (structural module type) a suitable style sheet is designed. Both, psychological and pedagogical as well as didactical aspects must be taken into consideration. The design of

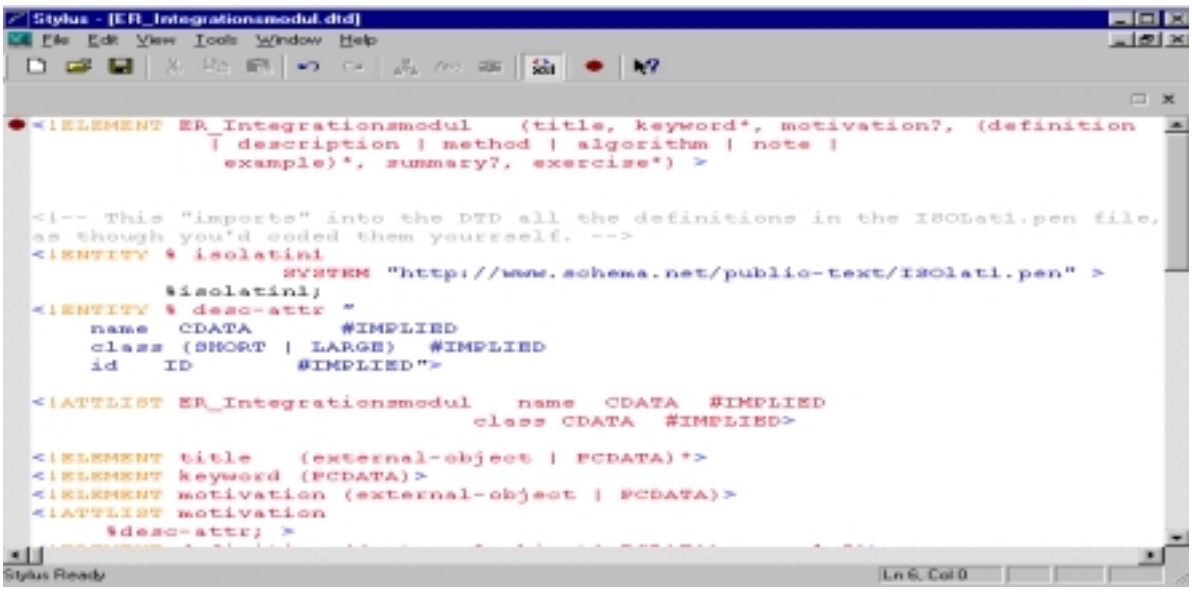

Figure 5: integration-module type: The integration module type of the subject area ER-Modelling is modeled through the container element ER-Integrationsmodule. A learning module of this type consists of the components: title, keyword, etc.. It also determines that its instances have one title, followed by one or more keywords and at most one motivation. Instances of this type can also include the components: definition, description, method, algorithm, note and example that can be combined in any number and in any order. An instance of this type can end with at most one summary and zero or more exercises.

the style sheets requires creative and artistic skills. Interface development is user-centered process [20] that iterates between design, implementation and evaluation by the user (see figure 2). The result of this phase is a style sheet for each structural module type.

6.2.4. Course design

In this phase learning modules are combined into course units. A course unit defines navigational structures between modules that represent a specific view. Again it is a configuration task, but this time not on the module level but on the level of a module set, i.e., modules are dealt with as a unit, and are no more configurable. Examples for navigation structures are: explorative navigation, e.g., searching in a module-repository, guided-tour, glossary, hierarchical organisation of the learning contents such as trees, etc..

To enhance reusability it is conceivable to define templates for various kinds of navigation structures. Authors of learning units need only to choose a suitable template and construct bridges (see section 5.3.4) depending on the chosen structure. Every thing else is done automatically by the authoring system. A system that implements a similar concept is GENTLE [9,10].
Results of this phase are:
Course units specification: the specification includes the modules that belong to the course unit, and their dependencies as they follow the logical dependency graph.
A navigation-structure template: The design or the selection of a suitable template for the desired navigation structure.

6.3. Content production (implementation)

This part of the development process deals with the implementation of the instances specified in every design phase. Therefore, there are three implementation phases on the module level and one implementation phase on the course level (see figure 2). In the following we discuss the different implementation phases. Discussing the implementation phase after we completely discussed all the design phases does not mean that the implementation can only begin after all design phases have been concluded. We suggest to follow each design step with its implementation step.

6.3.1. Implementation: integration module

Input for this phase are the results of the content-based design phase. For every identified module in the content-based design phase, an implementation process as shown in figure 9 has to be performed with the following steps:
Step 1: input for this step is the module specification. With the help of the description part of the module specification, implementors can search for the module in a module repository. If the search result is positive, the module may still not meet all requirements. In this case, the module will be adapted so that it meets the missing requirements, too. Otherwise, the module content has to be implemented. This is done by performing a development process for every module subcomponent found in the module specification. The development process of module subcomponents (i.e. module elements) is very similar to the development process of the modules themselves (see figure 9). Module subcomponents are searched in repositories of multimedia learning elements, such as ARIADNE [3]. If it is not found, then it has to be implemented using a suitable tool such as an animation-, text-, audio- or video-construction tool.
Step 2: after all subcomponent implementations are available, these components are assembled to create a module instance of type integration module.

6.3.2. Implementation: structural module

By the transition of an integration module to a structural module, subcomponents that still have the same specification are to be adapted from the integration module. New subcomponents have to be implemented as described above.

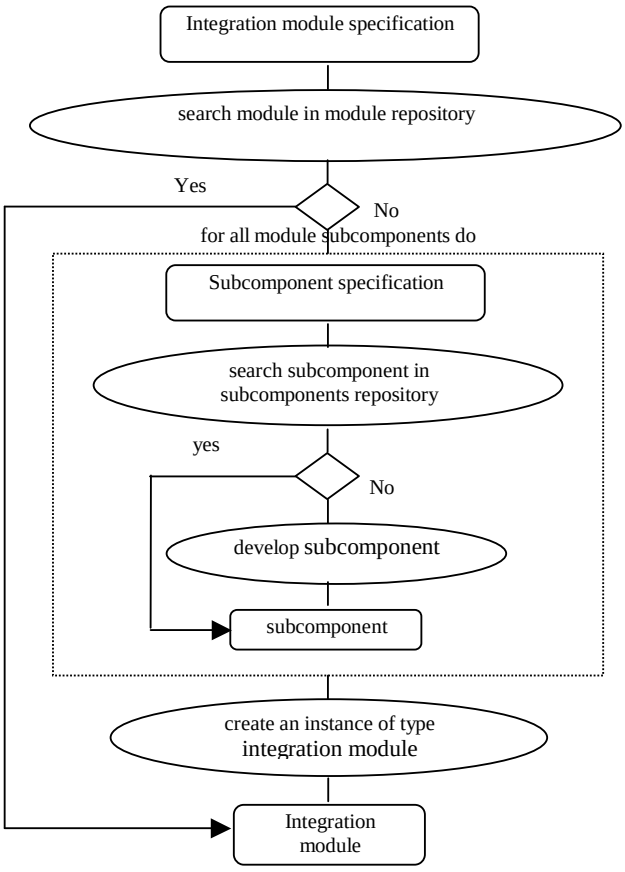

Figure 9 Implementation of integration modules

6.3.3. Implementation: presentational module

Presentational modules are automatically created by applying the style-sheets to the structural modules.

6.3.4. Implementation: learning units

Using the results of the design phase in section 5.2.4, authors need only to fill the placeholders in the designed navigation templates with presentational modules and eventually bridge them together. Because modules do not include any context-specific information such as "for further information see module B", depending on the chosen navigation-structure (e.g. guided-tour), context-bridges between the modules have to be constructed. Context-bridges include specific information about the context, where the modules are used. Sourcing out the context-specific information from the learning modules and placing them in context-bridges is very important for enhancing the reusability of modules.

7. Summary

In this paper we presented a concept that open a potential for the cooperative development of reusable multimedia courseware. Our approach is strongly based on modularization. It suggests to divide the learning materials into semantical units, the so-called learning modules, which allow to structure learning topics into semantically meaningful units, that may be (re-)used in various courses. We propose to organise the development of learning modules on several abstraction levels. These levels allow to abstract from author-specific characteristics, such as didactical and pedagogical concepts, from student-specific characteristics, such as learning form or background, and from context and learning type. Additionally, our approach makes use of the type concept to improve the reuse and shared use of concepts among modules.

As a next step, we plan to evaluate the practicability of our approach with a more sophisticated case study. Ultimately, we hope to develop an integrated cooperative authoring environment that supports all development phases of our concept.

In future work we think about integrating this development process into a learning environment, so that students will be supported to combine learning materials as individual views.

Acknowledgments

Our work is motivated and influenced by the ViKar subproject 2.3, especially by our discussions and cooperative work with Yue Chen, Müge Klein, Daniel Sommer, Klaus Gremminger, R. Krieger, and Wolffried Stucky.

References:

[1] D. Boles: Erstellung multimedialer Dokumente und Anwendungen: Verfahren und Werkzeuge. Workshop Software-Engineering für Multimedia-Systeme im Rahmen der GI '97-Jahrestagung, 1997.

[2] Virtueller Hochschulverbund Karlsruhe (ViKar), http://vikar.ira.uka.de/

[3] ARIADNE, http://ariadne.unil.ch/

[4] EDUCAUSE IMS, http://www.imsproject.org

[5] W3C: Extensible Markup Language (XML), http://www.xml.com/axml/axml.html

[6] IBM Learning Space; http://www.lotus.com/home.nsf/welcome/learnspace

[7] Oracle Learning Architecture, http://ola.us.oracle.com

[8] WebCT; http://www.webct.com/

[9] GENTLE, http://wbt-2.iicm.edu/

[10] HYPERWAVE, http://www.hyperwave.de/

[11] Garzotto, F., Paolini, P., and Schwabe, D.: HDM A model-based approach to Hypermedia application design. ACM Transactions on Information Systems, 11, 1 (Jan. 1993), 1-23.

[12] F. Isakowitz. A. Stohr, P. Balasubramanian; RMM: A Methodology for Structured Hypermedia Design. Communications of the ACM, August 1995, Vol. 38, No. 8.

[13] D. Schwabe, C. Rossi: The Object-Oriented Hypermedia Design Model. Communications of the ACM, August 1995, Vol. 38, No. 8.

[14] Erik Duval: An Open Infrastructure for Learning – the ARIADNE project. http://www.cs.kuleuven.ac.be/cwis/departement/personeel/

[15] Hyperskript project: http://iug.uni-paderborn.de/hyperskript

[16] S. Lewis: The Art and Science of SmallTalk; Prentice Hall, 1995.

[17] D. Geary: Graphic Java 2, Mastering the JFC; Prentice Hall

[18] E. Wilde: Wilde's WWW, technical foundations of the world wide web; Springer 1999.

[19] Eve Maler, Jeanne El Andaloussi: Developing SGML DTDs, From Text to Model to Markup; Prentice Hall, 1996.

[20] K. Ateyeh: Generic Graphical User Interfaces for ODMG Object Databases, Diplomarbeit, Universität Karlsruhe und University of Manchester, November 1998.

[21] K. Ateyeh, J. Mülle, P. C. Lockemann: Modulare Aufbereitung von multimedialen Lerninhalten für eine heterogene Lernumgebung. Bericht 1999-17, Fakultät für Informatik, Universität Karlsruhe, 1999.