Introduction:
Trends in the Design and Production of Computer Based Learning Material

Ton de Jong1 & Luigi Sarti2

1 Faculty of Educational Science and Technology, University of Twente, The Netherlands; 2 Istituto Tecnologie Didattiche, Italy

Nowadays courseware development is evolving from the status of handicraft activity to that of industrial process: the requirement that quality and cost-effectiveness should be ensured by the adoption of formal models, explicit methods and well-assessed tools is growing stronger and stronger. Software engineering underwent such an evolution years ago: it was a long and complex process that started with the definition of the first software life-cycle models and that is still in progress with the development of modern CASE environments.

Similarly to software, quality and effectiveness of courseware can only be achieved if the development process is modelled so that each activity can be supported at both the conceptual and technical levels. Long and expensive phases need optimisation through the development of proper standards and techniques. Reuse of existing artefacts and collaborative approaches are considered crucial aspects of the authoring activities.

Unfortunately, today’s off-the-shelf authoring tools seldom meet the above mentioned needs. Current commercially available authoring systems are most of the times based on a model of the development cycle that is either too generic, incomplete, or simply inadequate; as a consequence, high level design phases are usually neglected; reusability is poorly supported; representation and quality standards are far from being considered.

For these reasons research on better models, methods, and tools for designing and producing courseware is bound to have a crucial impact on tomorrow’s authoring systems. All the papers in this book discuss research projects that aim at the development of new ways to support the authoring process. The main focus in the present book is on design and production of courseware material, which are considered to be two critical activities. For these activities we propose the following two definitions:

- the design of the courseware material essentially means the definition of the domain content, context of usage, intended outcomes and instructional strategies for the courseware under development: in short, the ‘educational shape’ of the learning material;
- the production of the material refers to the materialisation of the decisions that have been taken in the design. Here, we can consider a variety of specific
activities, including the actual creation of multimedia material, the synchronisation of time-dependent components, and the organisation of the surface aspects of the learner interaction.

It is sometimes difficult to draw a clear line between these two aspects. For instance, the choice of the media used to present the material is usually related to the production phase, but might as well be considered at design time, given its strong links with such pedagogical aspects as the learner connotation.

Two classifications of papers in this book

Most of the papers in this book have something to say on both design and production, but some have a stronger design emphasis (Grandbastien & Gavignet; Mispelkamp & Sarti; Tait), some accentuate the production (Ulloa; Benamou & Celentano; Busch et al.; Flärtel), whilst others (Barker, van Rosmalen, de Jong et al.) pay balanced attention to design and production. Verreck and Weges, finally, take a broader view when they present a general framework (the Common Training Architecture) for describing courseware itself, the design, production and delivery process, and the context in which the courseware should function and in which the distinguished processes take place. In this way, CTA strives towards standardisation that should contribute to a higher level of “interoperability, portability, and ease of use”.

A second distinction that runs through this book is the one between ‘general’ courseware and simulation-based training material. This distinction is based on an essential characteristic of learning material that involves the way in which information on the domain is present. In one class of learning material this domain information is included in a ‘non reactive way’, and this is traditionally denoted as tutorial. In tutorials domain information is presented in some way to the learner (for example in (hyper)text, graphics, or demonstrations); in simulations a model of a domain is present, this model reacts to inputs of the learner, and the learner is expected to infer the characteristics of this model. Most of the papers in the book have to do with the more ‘tutorial’ type of approach, though some of them include simulations as a possible component of courseware (Benamou & Celentano). Four papers (Härte; Tait; van Rosmalen; de Jong et al.) are specifically dedicated to simulation based learning material. Also here, however, combinations of simulations with direct presentation of information appear (see e.g. de Jong et al.).

Emerging topics on design and production

One ‘design topic’ that emerges from the papers is the use of ‘basic’ entities for the design of learning material. These entities are called ‘learning units’ (Grandbastien) or ‘units of learning material’ (Mispelkamp & Sarti; Benamou & Celentano). Each unit (or “didactic module” as Benamou & Celentano describe it) has a specific
character. Grandbastien and Gavignet give as examples "presentation, exercise, review, and problem". Mispelkamp and Sarti relate the concept of unit of learning material to pedagogical goals for which they distinguish 'knowledge', 'understanding', and 'application'. Benamou and Celentano restrict the use of units of learning material to what they call a 'program oriented paradigm', which in fact is the tutorial approach described above, and, as Mispelkamp and Sarti do, they relate this to the specific goals in which also a well defined part of the domain is included.

In the area of simulation based learning we find a similar approach. Van Rosmalen describes the use of "Instructional Simulation Objects (ISO)", which are "basic educational units". Van Rosmalen distinguishes five different ISOs: demo, task, hypothesis, experiment, and exploration. De Jong et al., following an object oriented approach, describe what they call "basic building blocks". These basic building blocks are the "classes" an author can use for creating a simulation model and for instructional support around the simulation. In the SMISLE system, that is described by de Jong et al., the units, or basic building blocks, are grouped in libraries that are related to different functions in the resulting learning environment.

The choice for the type of units and their linking is a second topic in the design area. Units of material can be linked in many ways and different authors define different sets of relations. Grandbastien and Gavignet distinguish five different types of links between units of learning material which are divided into static links (links in which the learner's evolving knowledge is not considered) and dynamic links (in which the learner's developing skills are taken into consideration, for example whether a learner has passed or failed an exercise assessment). Mispelkamp and Sarti present the most elaborate typology of links by giving 17 types of link (though these links only zoom in on domain concepts).

A typology of links is one of the inputs for the final sequencing of units, which is determined by the instructional, or pedagogical, strategy that is chosen. Grandbastien and Gavignet give three of such strategies (that they label "meta-knowledge"), for example a strategy in which the presentation of information precedes exercises. Benamou and Celentano propose four different educational strategies of which the "program oriented strategies" is one. In this strategy the sequencing of units follows the pattern "presentation, interaction, test, and action". Barker presents an overall instructional paradigm that is supposed to be an "optimum mix" of several other learning paradigms. Barker calls his approach the MAPARI paradigm. This paradigm entails the following stages in the learning process: "mimicry, apprenticeship, practice, assessment, refinement, and improvement". Obviously in each of the stages different basic learning actions will take place. Tait, whose work is in the simulation area, divides the learning process into three phases with associated specific learning process: planning/prediction, performance/observation, and debriefing/feedback. Van Rosmalen uses what he calls "instructional plans", domain specific descriptions of how to reach a specific educational goal. In de Jong et al. authors are provided with instructional guidelines which help them to decide, for
domains and learners with specific characteristics, which instructional support measures should be available to a learner. The author can realise this by attaching so-called "enabling conditions" to instructional support measures.

An important issue in the design area is the level of support for authors. Traditional authoring tools essentially do not provide support that exceeds production techniques which centre around the provision of ‘flow charting’ support. The introduction of ‘units of learning material’, or ‘building blocks’ makes that authors now are supported by abstract (or even domain specific) templates, that they can use for creating their own courseware material. Support in this case therefore moves towards real ‘design’ support, since authors are offered ‘educational’ templates. This idea also translates into another new topic in design and production, which is the reuse of existing material. For efficient reuse, it is necessary that existing material is divided into smaller chunks, that have some generality and that are labelled with relevant characteristics, as is done for the above mentioned units of learning material and building blocks.

For both the design and the production of the learning material one of the new and emerging ideas is that of collaborative and distributed authoring. Uloa’s paper pays relevant attention to these aspects, depicting a scenario where a number of authors with different roles and from geographically distributed sites collaborate in the development of courseware. The co-authoring services offered in OSCAR include: collaboration tools, supporting group communication and sharing of information; co-ordination tools, supporting the organisation and management of projects; codecision tools, allowing remote group decision making; reuse tools, fostering the reuse of existing artefacts. Reuse is also a primary concern of the approach proposed both by Grandbastien and Gavignet, and by Mispeikamp and Sarti. Here, a sharp separation is recommended between the conceptual representation of the knowledge to be taught and the actual pieces of learning material the learner will interact with; besides, in both projects a strong distinction is drawn between the contents to be conveyed to the student (i.e., what to teach) and the instructional strategies used (i.e., how to teach). In both cases, the decoupling facilitates the co-operation between different authors and the reuse of existing components. Although not primarily concerned with distributed authoring issues, the emphasis de Jong et al. put on the provision of libraries of building blocks in the SMISLE toolkit constitutes another step towards the adoption of reuse techniques in the courseware development process. A number of aspects in the work of Benamou and Celentano can be related to collaborative authoring: here, authors can play a variety of roles: editor, teacher and course designer; specific tools are provided for such activities as resource management and session management; one of MATHESIS’ objectives is to provide an educational server accessed through local area networks and ISDN. The main objective of the COSYS project, described in Busch et al., is to pilot a distributed course production and delivery system. The co-authoring scenario includes such activities as project initiation and management, instructional design, production
planning, and material creation, editing and composition; co-authoring is supported through on-line access to a database of raw material and help facilities, and through a variety of communication facilities.

As we indicated above, courseware can be classified according to the nature of the domain that is present: simulation-based or not. Related to this issue are the formal techniques that are chosen in the different projects for describing the domain content of the courseware. Grandbastien and Gavignet propose an associative network where nodes are concepts of the domain subject area and edges represent either hierarchical relationships between concepts or such pedagogical knowledge as pre-requisite links. An interesting point is that the granularity of the network is determined by the assessment activities: the concepts of the network are the most elementary items for which the teacher requires assessment. Mispelkamp and Sarti support the usage of associative networks for a variety of purposes. The domain model, independent of specific learning goals, provides a reusable description of an area of knowledge; nodes are untyped, edges represent relations of various types: hierarchical, aggregational, temporal, spatial, causal, and related to activity. By applying learning goals to the domain model an author derives the content representation, another associative network where the nodes are typed according to a knowledge vs. skill taxonomy, and the edge type set is the same as for the domain analysis, plus the pre-requisite link type. Instructional object types are also organised into a formal hierarchy. In its concern for collaborative authoring and reuse models OSCAR explicitly represents all the actors and objects involved in the courseware development process as a network of classes. The design and production activities can therefore be formally modelled in terms of roles, functions, messages and rules. Strong inheritance hierarchies enhance the retrievability and reusability of artefacts.

The simulation contributions propose the adoption of a variety of specific graphic formalisms for the representation of relevant entities. Tait applies extended bond graphs to represent the functionality of physical systems as energy flows. Van Rosmaleren describes a hierarchy where a course is decomposed into learning goals, then into instructional plans and finally into instructional objects; such a hierarchy is explicitly represented in the system, and is used both by the authors in the design phase and by the learners as a navigation tool in the courseware space. SAM also represents a simulation model as a concept network at a variety of abstraction levels; instructional simulation objects are graphically represented as flowcharts. de Jong et al. use bond graphs to model a system in terms of energy flow, and extended petri nets to model both discrete events and operational expertise.

A final topic within the production area is the use of existing general applications (such as widely available text processors, or drawing tools) or the development of newly designed and dedicated tools. In the case study described in Barker’s contribution existing commercial tools are used in the framework of the proposed methodology. The OSCAR architecture described by Ulloa allows for the integration of external tools in the authoring space: Multimedia Toolbook™, Generic Tutoring
Environment (GETE) and Word for Windows™ are three examples of external applications included in OSCAR which required some customisation work to allow synchronous and asynchronous co-operative activities. Other kinds of tools, such as the Common Information Space Browser, required specific development. Härte's work lists a number of dedicated tools and final applications that have been developed in the COLOS project; emphasis is also put on the fact that 'experience shows a great reluctance of many teachers to the use of software in their teaching, especially if it is a foreign, 'out-of-house' product'. As a consequence of this approach, COLOS devotes significant effort to the provision of instruments that let non-programmer authors develop their own simulations. Van Rosmalen's project features a slot-based architecture where authors fill the slots of their SAM environment with either SAM-compatible applications or SAM-developed modules. A specific module is provided, which handles control and communication between individual tools; a package-independent command language is defined, to hide the complexity of platform—application-dependent technical details. The list of external, SAM-compatible applications includes Authorware™ and Microsoft Excel™. Editing of the material is supported in MATHESIS by a project-specific version of a commercial authoring tool, Apple Media Kit™, as well as by other external graphic and animation editing programs. The integration of external tools is also possible at student-time, to call and use them from within the courseware. The pilot production environments described in Busch et al. implement the Generic COSYS Concept using a wide set of commercial tools and formats, including Lotus Notes™, MS-Access™, FrameMaker™, Ventura Publisher™ and Windows™ Help.

Overview of the papers in this book

We conclude with brief descriptions of each of the chapters in the book.

Barker's work describes an approach to support the design and implementation of courseware systems based on a definition of interactive learning which attempts to mediate across different theories of learning, ranging from behaviourism to constructivism. Barker presents ten basic perspectives of learning design, as formulated in the LLDIC project, that each author should consider when creating courseware. The contribution also presents a case study where the proposed model and techniques are applied to the field of foreign language learning; a controlled evaluation study allows for a critical appraisal of the effectiveness of the approach.

ECSD, the generator of learning environments proposed by Grandbastien and Gavignet, consists of a formal model to describe the subject domain and a set of tools which enable the author to associate the elements of the domain structure with existing learning units, characterised by the definition of pre-conditions (elementary items) and post-actions (learner's results). The model also addresses such issues as pedagogical objectives, dynamic evaluation of the learner's performance and instructional strategies.

Mispelkamp and Sarti describe the DISCourse approach to courseware design, including those aspects like learner modelling, instructional design and domain
INTRODUCTION

representation. Aimed at the systematic support of all the phases of courseware development, DISCourse (Design and Interactive Specification of Courseware) addresses the representation of the subject domain at three levels: domain analysis, learning goal analysis and content analysis. Particular attention is paid to reusability issues. Specific tools supporting the author in these activities are also described.

Benanou and Celentano describe MATHESIS, a project aimed at the definition and development of a stand-alone workbench for both the author and the student. The main features of the MATHESIS proposal are its flexible attitude towards educational strategies, its strong support of multimedia and hypermedia functionalities and the independence of its architecture from the underlying hardware platform. External tools can be integrated in the workbench at various levels of binding.

OSCAR (Open System for Collaborative Authoring and Reuse) addresses collaborative and distributed authoring of multimedia training materials. Ullon sketches an application scenario where authoring resources are distributed over both local and wide areas, with the aim of fulfilling the requirements of uniformity, quality, productivity and reliability for the course development process. OSCAR is based on an open system architecture and provides multimedia communication facilities, a common information space allowing co-operation between different tools and actors, and high level services to support collaboration, co-ordination, codecision and reuse.

The main objective of COSYS is to pilot a distributed course production and delivery system. Busch et al. describe that COSYS aims at users such as publishers, producers of courseware, distributors, authors, etc. A central concept in COSYS is that users of course material that is filed in a database should be able to tailor the material to their own needs. COSYS offers a host of services that at the most detailed level translate into for example stylesheets for the presentation of material.

Tait reports on the part of the DISCourse project (of which the general part is presented in Mispelkamp and Sarti) that takes care of simulation-based learning environments. In the DISCourse project a systematic methodology is developed for the design and production of a simulation, based on an abstract representation of the model which drives the simulation. For creating the underlying model DISCourse relies on bond as a formalism and also a distinction between a 'pure' domain model (created by a modeller) and a domain model suited for education (created by an author) is made. DISCourse provides additional support to the learner. This support comprises: proposing useful tasks (which can be manipulation or diagnosis tasks), encouraging the stating of hypotheses (a set of propositions) and expectations (represented in graphical form), providing explanations, answering learners’ questions, and making suggestions for subsequent investigations. The main application area of the DISCourse simulation system is in the field of medicine.

De Jong et al. present the SMISLE (System for Multimedia Simulation Learning Environments) project. The aim of this project is to build a general authoring environment for simulations together with instructional support. Four specific types of instructional measures were selected: progressive model implementation,
assignments, explanations and hypothesis scratchpads. In SMISLE authors create integrated simulation learning environments by selecting 'building blocks' from libraries and by tailoring these building blocks to their needs through the use of dedicated editors. In this respect the design, (e.g. devising instructional support) and production (e.g. creating this instructional support) are united. Additionally, authors are guided through the authoring process by a methodology and they have access to instructional advice which provides them with ideas on which instructional measures to apply.

The SAM (Simulation And Multimedia) project is described by van Rosmalen. The aim of the SAM project is to specify and develop a modelling, authoring and learning framework for simulation-based learning based on external software tools. SAM, therefore, stands for the integration of existing tools, but adds specifically designed SAM tools. At a technical level the integration of existing and developed tools concerns the communication and control between the tools reflected in the creation data exchange protocols. SAM guarantees the conceptual integration through the use of so-called Instructional Simulation Objects (ISO). ISOs are building blocks that allow for different levels of freedom ranging from fully system guided task exercises to learner controlled experiments. With help of a graphical editor the author writes a script both controlling SAM designed functions, e.g. monitoring or exchange of data, and external supplied functions, e.g. running a simulation model or a courseware. To the learner SAM will supply additional tools such as e.g. a multi-media notebook.

Härdel gives insight into the COLOS (Conceptual Learning of Science) project. This project builds on a large number of local initiatives and uses as such a bottom-up approach. The common interest of all sites is the use of simulations to help students gain basic concepts in science by reducing the mathematical overload so often characteristic for traditional science instruction, to provide the possibility of easy experimentation with possible change of time scales etc., and to support learners' knowledge acquisition process by offering spatial, 3-dimensional, representations of phenomena. The project develops a core of authoring instruments that allows a non-expert programmer to develop models forming the basis of simulations together with the necessary interaction part. The paper describes this set of tools. Secondly, the project provides authors with 'ready made', but still customisable, applications in specialised fields such as neural nets and integrated circuits. The applications created in COLOS are meant for teachers to be used in their lectures, but may also be given to students directly.

The CTA project, finally, as it is presented in the paper by Verreck and Weges has different objectives than the other projects described in this book. CTA is not concerned with developing methods that can be used for design and production, but gives a higher level framework for analyzing and formalising design and production and the methods that are used themselves. The final goal of the CTA project is contribute to harmonisation and standardisation of technologies, systems, and infrastructures for flexible and distance learning.