A Generator and a Meta Specification Language for Courseware

Bernd Gaede (gaede@forwiss.de)
Bavarian Research Center for Knowledge-based Systems,
Haberstrasse 2, D-91058 Erlangen, Germany

Herbert Stoyan (hstoyan@informatik.uni-erlangen.de)
Chair of Artificial Intelligence, Friedrich Alexander University of Erlangen-Nürnberg
Am Weichselgarten 9, D-91058 Erlangen-Tennenlohe, Germany

Abstract: Courseware has to reflect the learning situation coined by contents, learning goals, characteristics of the learner, technical conditions etc. The efficient creation of appropriately designed courseware relies on flexible representations of reusable instructional material and learning circumstances.
Contradictory approaches of different instructional design schools give little advice in case of design alternatives: The support gap for justified design decisions ranges from underlying instructional theory to ergonomic layout. Therefore, we separated factors influencing courseware design from design parameters and present the Instructional Material Description Language IMDL for their respective description. A simple rule language serves to express dependencies between influence factors and design parameters. Rule sets represent prescriptive design statements based on instructional design theory which are open for validation.
A toolset enables efficient courseware production based on IMDL, e.g. an interpreter engine for rules produces complete courseware specifications and a generator outputs source code for specified courseware based on a framework of instructional Java-Beans™.

Introduction

The shift to the information or knowledge society is accompanied by the need for continuing education. The corresponding need for courseware grows with the gap between demand and otherwise deliverable supply of education. The efficient development of courseware is thus an important goal. Proprietary multimedia authoring tools like Authorware™ or Tooolbook™ have been extended to meet the specific needs of courseware authors, but we don't think that WYSIWYG-editors provide an appropriate methodological support for the systematic development of courseware. WBI (Web-based instruction) is taken for another remedy because it makes development faster for a wide area of applications. The simple and standards-based technology of the WWW has proven fruitful and is easily to learn and implement when persisting bandwidth limitations don't obstruct its use. However, our point of view is that they may help non-programmers to build multimedia applications but tend to distract their attention from didactic and content related aspects. The first problem is thus lacking instructional design knowledge and tool support. Furthermore, attractive courseware that supports learning on demand has to be modular and designed to meet the targeted learners' needs and thus requires even more complex engineering models or methodologies like the ones arising from the area of knowledge management.
Instructional System Development (ISD) is concerned with the former mentioned problems. We interpret the term "instructional system" as any system to foster learning, whereas our work is concerned with development of instructional software only. ISD consists of the phases planning, development and implementation (use and evaluation) where we focus on the development phase that in turn consists of design and production. A pragmatic definition of *Instructional Design* (ID) that we agree with is given by Lowyck & Elen (1993) who say that ID is a discipline that connects descriptive research findings with instructional practice by (1) identifying design parameters based on results of basic research from cognitive psychology; (2) instruments these as rules, procedures and methods and (3) provides prescriptions for the development of instruction to optimise teaching and learning. The most ambitious approach to put this definition to practice probably is the automation of ID that Tennyson & Baron (1995) or more recently Spector & Muraida (1997) give a survey of.
Approaches from Software Engineering and Knowledge Management resulted in courseware standardization efforts like the *Architecture and Reference Model* by the *Learning Technology Standards Committee* of the

IEEE [http://ltsc.ieee.org]. Its working group for *Learning Objects Metadata* develops a metadata standard to describe learning resources. A meta modelling approach to the development and management of teachware based on re-use of contents and structure that uses metadata as a foundation of adaption is the *Passau Knowledge Management System* PaKMaS (see Brössler, Freitag & Süß, 1999). However, PaKMaS is not out to produce courseware but is a management system that transforms IM-models into hypertexts. On top of that, the modelling is done without didactical guidance or ID advice. Murray (1996) suggests *"ontology objects, which allow for the creation of representational frameworks tailored to classes of domains or tasks (…)"* for *Intelligent Tutoring Systems* (ITS).
We took up this approach and have built the *Instructional Material Description Language* (IMDL) based on XML. This multi-layer specification language consists of three parts to enable generic descriptions of courseware. IMDL-subsets serve to describe influence factors, design parameters and there dependencies. Base specifications of IMDL then simply describe actual material in terms of relevant influence factors (like content, structure, targeted learner and so on); complete courseware specifications become possible by the addition of a set of elements to describe design parameters. Elements of both subsets describe well-understood parts of the domain, whereas controversial aspects are ontologically separated; author-specific extensions. ProfiL, the *„**Pro**duktionssystem **f**ür **i**nteraktive **L**ernsoftware"* (German for "Production system for interactive courseware") first transforms base specifications into complete courseware specifications according to author-formulated dependencies (design rules) and finally generates according courseware based on a framework of instructional JavaBeans™.
In the following section aspects of courseware are analysed and summarized distinguishing between influence factors and design parameters before our approach to explicitly represent instructional design axioms is presented. Afterwards, the XML-based implementation of the approach as IMDL is roughly sketched. Another section addresses the architecture of ProfiL after clarifying the roles and relations of documents and document type definitions. The final section provides a brief summary, our conclusions and an outlook on future research and development.

Aspects of Courseware: An Analytical Approach

The description of courseware is not yet comprehensively possible due to its interdisciplinary and complex character. After all, various sets of metadata for learning resources can be found in the literature some of which (e.g. Learning Objects Metadata, IMS [http://www.imsproject.org] and ARIADNE [http://ariadne.unil.ch]) are standardized. None of these approaches provides vocabulary to describe didactical aspects, instead they are extensible for that purpose. We are convinced that this lack of common vocabulary indicates the general state of instructional design research, that is characterized by Seel and Dijkstra (1997): *"An analysis of traditional ID demonstrates that it is not possible to map results of empirical research on learning onto instructional planning in a one-to-one manner, but rather the transformation of a descriptive knowledge base into a truly prescriptive instructional theory seems to be still in its infancy."* The situation is rather characterized by anecdotical results that are hard to compare or to integrate within a sound theory of ID as concluded in Kerres (1998), Mandl & Reinmann-Rothmeier (1997) or Schulmeister (1997).
For that purpose we tried to approach these problems in a bottom-up fashion: We first analysed the state of the art of ID and current trends of courseware engineering to identify influence factors and design parameters of courseware. The latter have then been considered as dependent variables with respect to the former. In the following, we want to distinguish instructional material (IM) from courseware: IM are self-contained units of learning content, that are also called knowledge objects (e.g. by Merrill, 1997) or conceptual units (Brössler, Freitag and Süß, 1999) in the literature. Absent contextual embedding and functionality make the distinction between the two. Based on this preliminary remark we can now take look at the influence factors of courseware design.

Influence Factors of Courseware Design

The inherently interdisciplinary development of courseware requires to take into account various different aspects. ID usually distinguishes between didactical assumptions (based on learning theory, media didactics, software ergonomics), contents and technical aspects. We pragmatically adapted this view by subsuming contents and learning objectives under instructional material. IM is thus particularly characterized by contents that we model as concepts, that are initially independent and that can be annotated with arbitrary learning objectives, e.g. according to Bloom, Ausubel or Gagné. Our focus is on cognitive and motivational affective objectives, since we don't expect computers to be universally appropriate, e.g. for psychomotoric training.

Further aspects of IM are its mediality and granularity whereas its state (with respect to certainty or consensus about contents) and difficulty can be coded as metadata.
Structure of IM is another essential feature that we model as concept net. Link types can be added based on their graph-theoretical properties. The overall complexity can be expressed via metadata.
Learner properties are treated in such a heterogeneous way in the literature that an exclusive commitment would be pointless. We thus realized a common and flexible division into a (domain) knowledge model and a learner attribute model. The former is implemented as simple overlay model, whereas the attribute model is made up of descriptive primitives, among which appropriate subsets can be selected according to the learning situation. If patterns within the space of possible instantiations should be re-used this can be achieved via specific extensions. Overlay-learner-models can be re-used in the same way, e.g. by approximatively characterizing a pupil with respect to his previous knowledge by his grade. Further primitives are dedicated to media preferences (verbal – textual or auditive - or visual type), motivation (high, medium, low) computer literacy, availability (in terms of time to learn), environment, age, cultural predisposition (that sometimes insinuates analytical or synthetical learning preferences) and further learning styles, e.g according to Kolb or Keirsey.
Another important component of most learning systems is assessment. Its application ranges from self control over feedback and adaption to certification. Authoring appropriate tests is a complex subject matter on its own that goes beyond the scope of this article. To automatically evaluate tests and integrate them into courseware, at least goals (e.g. as a threshold for variants mastery and normative) have to be specified as metadata.
The sketched selection of influence factors is not complete and doesn't take into account important aspects like multilingualism, handicapped learners or psychomotoric learning objectives. The building of this ontology of influence factors certainly is a process that can hardly ever be considered to be completed.

Design Parameters of Courseware

This section presents design parameters of courseware that exist dependent on or independently of the above influence factors. Attention is also paid to possible ranges of values for these influence factors. We have derived an abstract generic model of courseware from a survey of available products and research prototypes. In addition, this model tries to integrate results of multimedia software engineering (see Gaede & Schneeberger, 1998) with the domain of Intelligent Tutoring Systems (ITS) and learning theory. Multimedia software should be built in a modular fashion that separates structure, contents, layout and functionality. ITS research contributes the separation of domain, learner and didactical module as well as user interface. Learning theory and media-didactics don't actually propose a specific courseware architecture but the latter offers (partially general) rules to select, arrange and layout media. Learning theory with respect to instructional design is considered in the next section.
On the basis of this model we have localized possible differences and identified them as variables with exemplary values. These design parameters are related to all aspects of courseware again, which are: content, structure, learner, tests, and beyond this, to functionality. Additionally, metadata can be used according to specific requirements or established standards. The multi-layered specification design based on learning theory and media-didactics shall be outlined next.
From a didactical point of view, macro- and micro-strategies can be distinguished. Macro-strategies apply to sequencing of units and, in a more abstract way, to types of courseware (presentation, drill & practice, tutorial, and so on). Navigational options like a given degree of freedom or generally available access variants (sequential, hierarchical, guided tour or explorative) also belong to this sector. Micro-strategies relate to the selection of components (e.g. definition, explanation, summary, example, analogy, hint, feedback, question, answer, task) depending on their type and function. Selection of media and feedback variants plays another important role in this connection.
The design options with respect to contents are mainly made up of further selections. For example, the choice of IM can be done with respect to its difficulty or to previous knowledge of the learner.
The technical perspective is the easiest to handle though lots of factors play a role. Access (online/offline), if applicable bandwidth, target platform and many others fit in here. We tried to respond to related problems by choosing the largely platform-independent programming language Java and alternative generation of HTML code. However, the selection among different media representations can be made dependent on such aspects.

From Learning Theory and Didactics to Prescriptive Design Axioms

Now that we have presented primitives to describe influence factors as opposed to design parameters, it is quite straightforward to define design instructions in terms of relations between values of the former and the latter!

Nevertheless, it remains problematic to completely specify such dependencies, first, because of the obvious combinatorial explosion; and second, because of the mostly anecdotical research that can hardly be called empirical. The first can be tackled by abstraction, when describing primitives are hierarchically synthesized. The latter is a long-term goal of our approach, by making the usually implicit assumptions of ID explicit in terms of rules, that can be compared and evaluated.
Advocates of the tree main trends in learning theory, behaviourism, cognitivism and constructivism, have different views of learning (processes) from which again different design instructions for courseware are derived, not to mention the varying terminology across and within the ID schools. The at present prevailing constructivist community remains (like its elder relatives) vague in its conclusions. So states Gruender (1996): *„However, contrary to the claims of constructivism, nothing follows from this theory about what the best educational methods might be to help people to learn."* The ProfiL-methodology thus enforces the formulation of detailed, specific design rules by courseware authors. Subsets of corpora of such rules can be pooled and arbitrarily named. In doing so, a number of rules that describe courseware design according to Ausubel (1960) could be organized as set of prescriptive axioms and be given a name like 'Ausubel_for_Hypertext'.

A Specification Language based on XML: IMDL

We have chosen the platform-independent meta document description language XML *(Extensible Markup Language)* for the implementation of the specification language because of the standardization efforts and the promising development of the WWW. It enables extensible language definitions using DTDs *(Document Type Definitions,* whereas XML-Schema definitions offer extended possibilities but still have the state *W3C Working Draft)* and thereby meets our requirements. The expressiveness of DTDs is similar to formalisms like EBNF although certain aspects are complicated to represent. The following three sections give an overview of IMDL, the *Instructional Material Description Language,* by sequentially presenting its subset to describe influence factors IMDL-I together with the subset for design parameters IMDL-D, their respective extensions IMDL-IE and IMDL-DE, and the IMDL-R to build rules based on the former subsets. The layered architecture of IMDL further distinguishes between layers for core-, strategy-, model-, component- and layout-specification but is suppressed here for clarity and brevity.
We present the higher levels of both descriptive languages. IMDL-I consists of five modular parts to describe learners, learning-objectives, domains, reusable media-units and reusable framework components. Details are hidden in the files for the different modules. IMDL-D contains elements and attributes to specify didactical objects and structures and is extended by elements to describe abstract software components and layout.
Whereas we tried to identify and name influence factors and design parameters at a fine-grained level, we have to provide the means to efficiently use them in an author-specific way. Authors aren't forced to explicitly use primitives but may use or add their own hierarchical language extensions for further use. One example is given to illustrate this for each, IMDL-IE and IMDL-DE: When previous math knowledge of a 7th class pupil is modelled once with IMDL-I, the extension '7th_class_pupil' could be added to IMDL-IE and then be used instead. If a design pattern of IMDL-D consists of information presentation, a test and depending on its result repeated presentation or presentation of another topic and the author wants to re-use this pattern he can introduce a language extension called 'Drill&Practice' and in future use that element of IMDL-DE instead. These examples illustrate, that a set of best-practices of language extensions might be agreed upon by author communities and thereby result in a kind of standard representations of instructional patterns.

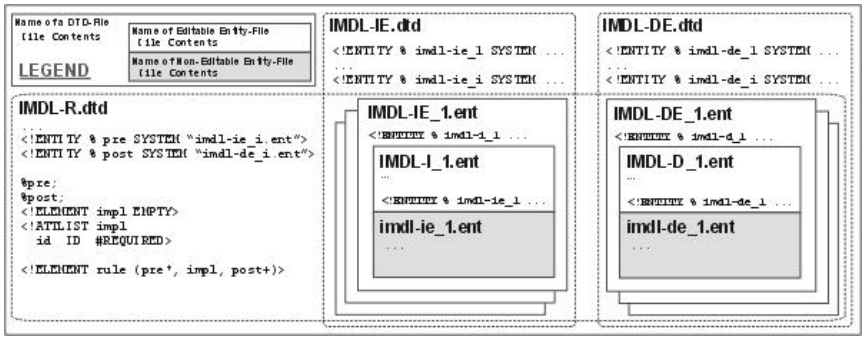

Figure 1: Relations of Document Type Definitions and Entities for Specifications and Rules

Language extensions are added in separate files and included into the common 'root file' via entities (see **Figure 1**). This has been necessary to avoid editing of the fixed language parts IMDL-I and IMDL-D. The root file isn't edible either and contains only entities to include these constant parts and entities to include the variable parts edited by the author. All language extensions are done by the help of DTD-editors that will be extended by more specific tools to support this task. Authors can thus access and re-use all elements and attributes of included entity files.

Prescriptive Description of Design Rules: IMDL-R

Given the XML-based IMDL, the Extensible Stylesheet Language (XSL) and its part XSLT (Transformations) would be the obvious choice to describe design rules. XSLT is a language for transforming XML documents into other XML documents. It uses patterns and templates for conditions and actions respectively. However, the way rules are processed by so-called XSL-engines is deterministic whereby application of a set of rules to a specification (an IMDL-IE document) yields exactly one target (IMDL-DE) document. Since our goal is to generate alternative variants in case of under-specified influence factors, we defined the (trivial) rule language IMDL-R and implemented an according interpreter. The DTD for the rules allows for elements of IMDL-IE as pre- and elements of IMDL-DE as post-conditions.
Inclusion is done via entities again, because the XML standard and according tools don't make it possible to include complete DTDs (i.e. with header). Therefore, the root-elements of IMDFL-IE and IMDL-DE are eliminated and the thus truncated rest is included unchanged.
We alternatively consider language elements as terms and work on a pre-processor that transforms specifications into first-order logic. Thereby, specifications of IM are to be seen as sets of facts whereas design rules become axioms. This enables the use of efficient theorem-provers instead of simple interpreter engines and lets sets of rules be checked for consistency and redundancy.

Architecture of ProfiL

We want to begin by illustrating (in **Figure 2**) the relations of the documents and document types described above before we sketch the architecture of the entire ProfiL-system. The DTDs IMDL-I and IMDL-D define how influence factors and design parameters are described at the lowest level respectively. They thus define the core language of ProfiL. Each of these two is included (1.) into the respective author specific DTD file IMDL-IE and IMDL-DE. IMDL-R lays down the syntax of rules to describe dependencies among them, what for essentially only variable symbols, quantifiers, sets and an implication symbol are introduced and the elements of IMDL-IE and IMDL-DE are assigned the role (2.) of pre- and post-conditions respectively. The IMDL-R.dtd is not editable but all elements an author added via extensions in IMDL-IE and IMDL-DE are available within rules.

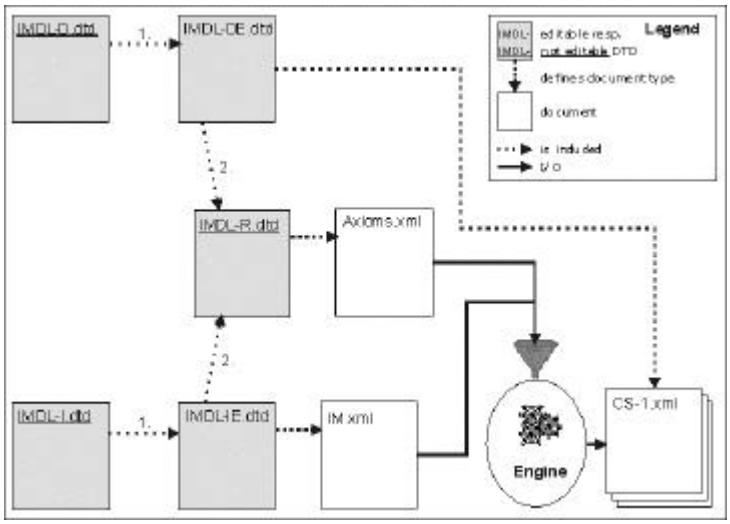

Figure 2: Relations of Documents for Specifications and Design-Rules

The document „Axioms.xml“ is written by an author or a group of authors. It lays down in the rule language IMDL-R specific design rules to be applied on specific conditions. The example document „IM.xml“ in turn describes actual IM with respect to contents and other influence factors. Both documents together are processed by the interpreter („Engine“) resulting in different variants of complete courseware specifications, here called „CS1.xml“ and following. To keep the illustration clear we have suppressed the multi-layer structure of the specification languages. Actually, language elements are distinguished as belonging to a core-, strategy-, model-, component- and layout-layer. Rules are accordingly applied in a multi-phase process.

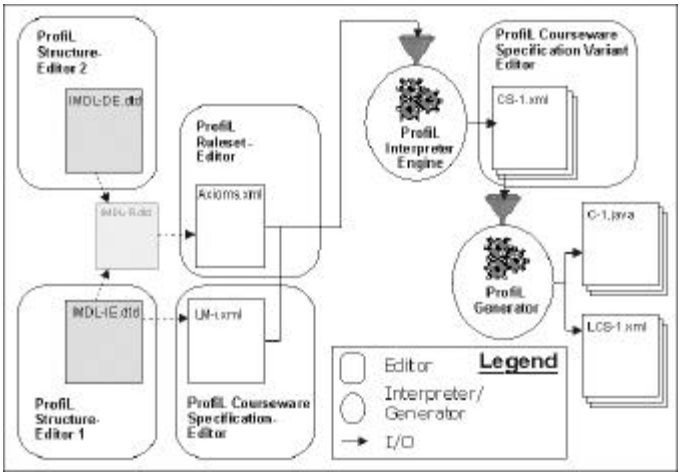

Figure 3: Tools and Document Flow within ProfiL

We are now going to consider the roles of documents within the overall architecture of ProfiL as illustrated in **Figure 3**. A standard editor has been integrated to edit the extensible DTDs IMDL-IE and IMDL-DE; for clarity we labelled it „Structure Editor 1“ and „Structure Editor 2“ according to its use. The non-editable DTD IMDL-R parameterizes the „Ruleset Editor“ that serves to author instructional design rules. The same XML-Editor (now labeled „Courseware Specification Editor“) is used to author specifications of IM when parameterised with the DTD for IMDL-IE. The interpreter known from **Figure 1** then generates several variants of courseware specifications („CS-1.xml“ and others) that can again be manually edited with the XML-Editor in the role of a „Courseware Specification Variant Editor“. This allows for review and selection of individually preferred variants and for individual design changes (exceptions) that shall not persistently influence the rules. Finally, the „ProfiL Generator“ is invoked with the selected specification(s) and produces Java source code („C-1.java“) that can be compiled into a executable application. The documents depicted below are called „LCS-i“ standing for „*Lazy Courseware Specification*“. The files contain either parts of the influence factor specification that haven't been fixed at design time or dynamic parameters that are expected to change during courseware use. By analogy with partial evaluation, these parts aren't evaluated immediately (during generation) but at runtime. An obvious example for this case is a learner's domain knowledge model that of course has to adapt to the extended knowledge during learning. If this is impossible or not sensible, several alternative variants are generated.

Summary and Outlook

The separation of influence factors and design parameters and the explicit description of their dependencies is an essential prerequisite to consolidate instructional courseware design. The use of the ProfiL-toolset and the proposed methodology enforces courseware authors to make assumed dependencies explicit. This enables validation and comparison of didactical alternatives. The framework-based generation approach makes use of the extensible specification language IMDL. This results in modular and re-usable instructional material leading to more efficient courseware production. The final fully automatic generation of source code based on instructional Java-Beans in compliance with ergonomic and didactical aspects results in further savings. The implementation of the presented system is almost complete; all components have already proved to work as prototypes.

However, there remain problems: The use of our tools is not yet easy enough (e.g. compared to the tool described by Ainsworth & Grimshaw, 1999) and abstraction via extensions alone doesn't seem to make

authoring easy enough either, though we expect improvements by collaborative ontology engineering as described by Staab (2000). Supposing that universally valid terminology is neither near nor realistic, partial domain- as well as pedagogical ontologies (see Mizoguchi Bourdeau, 1999) have already proven fruitful. The collaborative management and construction of ontologies by means of namespaces might thus help to ground courseware design in the spirit of Hannafin (1997). Another shortcoming of ProfiL is the renouncement of runnable domain models that make up the power of ITS but it is certainly possible to integrate such features after revising the framework that underlies the code generation. The actually implemented structural model already enables the automatic generation of simple tests. The lacking store of experience and the tedious initial specification of IM and design axioms are critical and at present prevent us from evaluating the authoring process let alone subsequent courseware use. It thus remains in question whether representing ID-knowledge as rules is acceptable to courseware authors. Early attempts towards programmed ID couldn't prevail, but we attribute this to lacking increase in value (e.g. the advice system of Elen & Stevens, 1993) that multimedia courseware has overcome. Moreover, these approaches where often committed to a single instructional theory like Frank's (1991) system ALSKINDI, that solely realized programmed instruction according to Skinner.

The quality and efficiency of courseware development according to the ProfiL-methodology can be further increased by preceding measures related to the authoring of instructional content. Guidelines for the didactical text production can help domain experts with this task. Templates can result in better structured content and can easily be added to the IMDL. Very important is the improvement of the graphical editors that are refined at the same time the example specifications and axioms are further developed. We thereby try to directly take into account specific requirements. The integration of theorem provers is only loosely coupled with the rest of the 'roadworks' but we expect the additional generation of explanatory documentation of it: Tracked chains of applied design rules promise to provide justifications for produced courseware.

References

Ainsworth, S.E. & Grimshaw, S.K. & Underwood, D.J. (1999): *Teachers Implementing Pedagogy Through Redeem* In:Computers & Education, 33 (2-3) pp. 171-187.

Ausubel, D.P. (1960): The use of advanced organizers in the learning and retention of meaningful verbal material Journal of Educational Psychology, 51, 267-272.

Balzert, H. et al. (1999): Studie über softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme. Technical Report, Forschergruppe Softec NRW.

Brössler, Peter & Freitag, Burkhard & Süß, Christian (1999): *Metamodeling for Web-Based Teachware Management*. In: P.P Chen, D.W. Embley, J. Kouloumdijan, S.W. Liddle und J.F. Roddick, *Proc. Intl. WWWCM`99 Workshop on the World-Wide Web and Conceptual Modeling in conjunction with ER`99*, LNCS 1727, Springer Verlag.

Elen, J. & Stevens, W. (1993): *A coach for instructional decision making* Paper presented at the Instructional Design SIG Symposium *Tools for instructional design*, 5th European Conference for Research on Learning and Instruction Aix-en-Provence.

Frank, H. & Meder, B.S. (1971): *Einführung in die kybernetische Pädagogik* Deutscher Taschenbuch-Verlag, München.

Gaede, B. & Schneeberger, J. (1998): *Generierung multimedialer Produktpräsentationen* In: WIRTSCHAFTSINFORMATIK Heft 1, Februar, 40. Jahrgang.

Gruender, C.D. (1996): *Constructivism and learning: A philosophical appraisal*. Educational Technology, 36 (3), pp. 21-29.

Hannafin, M.J. (1997): The case for grounded design: What the literature suggests about effective teaching, learning, and technology. Keynote presentation at the annual ASCILITE meeting, Perth, Australia.

Kerres, M. (1998): Multimediale und telemediale Lernumgebungen, Konzeption und Entwicklungen. Oldenbourg Verlag, München.

Lowyck, J. & Elen, J. (1993): *Transitions in the theoretical foundation of instructional design*. In T.M. Duffy, J. Lowyck & D.H. Jonassen (Eds.) *Designing Environments for Constructive Learning*, pp. 213-231. NATO ASI Series, Heidelberg: Springer-Verlag.

Mandl, H. & Reinmann-Rothmeier, G. (1997): *Lernen mit Multimedia.* Forschungsbericht 77 der Ludwig-Maximilians-Universität München.

Merrill, M.D. (1997): Instructional Transaction Theory: An Instructional Design Model Based on Knowledge Objects. In: Tennyson, R.D. et al (Eds.) Instructional Design. International Perspective. Vol. 1: Theory, Research and Models. Hillsdale, NJ; Lawrence Erlbaum Ass. pp. 361-394.

Mizoguchi, R. & Bourdeau, J. (1999): *A Multiple View Authoring Tool for Modeling Training Materials* AI Technical Report 5-99, I.S.I.R., Osaka University.

Murray, T. (1996): *Special Purpose Ontologies and the Representation of Pedagogical Knowledge.* In: Proceedings of the International Conference on the Learning Sciences, 1996. AACE, Charlottesville, Virginia.

Murray, T. (1998): Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. In: Journal of the Learning Sciences, Vol 7, No 1, 1998, pp. 5-64.

Reigeluth, Charles M. (1999): Instructional-Design Theories and Models: A new Paradigm of Instructional Theory. Lawrence Erlbaum Ass.
Schulmeister, R. (1997): *Grundlagen Hypermedialer Lernsysteme* Addison-Wesley, Berlin.
Seel, N. & Dijkstra, S. (1997): *General Introduction*. In: Dijkstra, S. et al (eds): *Instructional Design: International Perspective*. Vol 2: *Solving Instructional Design Problems.* Hillsdale, NJ; Lawrence Erlbaum Ass. pp. 1-13.
Spector, J.M. & Muraida, D.J. (1997): *Automating Instructional Design.* In: Dijkstra, S. et al (Eds.): *Instructional Design: International Perspective*. Vol 2*: Solving Instructional Design Problems.* Hillsdale, NJ; Lawrence Erlbaum Ass. pp. 59-81.
Staab, S. & Maedche, A. (2000): *Ontology engineering beyond the modeling of concepts and relations.* In Proceedings of the ECAI'2000 Workshop on Application of Ontologies and Problem-Solving Methods. IOS Press, Amsterdam
Tennyson, R. D. & Baron, A. E. (1995): Automating instructional design: An introduction In Automating instructional design: Computer-based development and delivery tools. New York: Springer-Verlag pp. 1-10.