MTEACH: a simple framework for didactic and context-based hypermedia

Pier Luca MONTESSORO, Davide PIERATTONI, Elio TOPPANO

Abstract—The MTEACH authoring framework is based on a methodology supported by a language and a compiler, allowing the author of a didactic multimedia production to work at a higher level than conventional authoring tools. Starting from a formal description of the lessons, the running code of the multimedia application is generated in standard formats, and provides a consistent and easy-to-navigate hypermedia structure.

Automatic link generation provides reusability of the multimedia material after it is organized in the lesson structure. This approach will allow authors to create different context-based presentations starting from the expected skill of the target users. Advanced indexing schemes will be further available for dynamic and interactive hypermedia generation.

The MTEACH research project is also developing a stable workspace for almost real-time multimedia production. Interfacing the teacher’s console with a multimedia processing unit, lessons and knowledge models can be directly edited and assembled in a final hypermedia product. This represents also a fast prototyping environment for experiencing advanced distance learning models and methodologies.

Index Terms--Distance learning, hypermedia, authoring, real-time production.

I. MAIN MOTIVATIONS AND OBJECTIVES OF THE MTEACH PROJECT

Facing the design of a didactic-oriented authoring framework, software developers should identify the correct guidelines for the final success of the project. These are strictly dependent from the target of the authoring tool; both author's and end user's needs have thus to be satisfied, starting from an investigation on their practical requirements.

Didactic represents a natural application field for multimedia systems and authoring techniques. Traditional live lessons are basically "multimedia": the teacher talks, while explaining slides and pictures, showing objects and so on. With some improvements and adaptations, live lessons can be placed on an interactive CD-ROM or a Web site, as a support for almost real-time multimedia production. Advanced models too require the immediate communication medium represented by a teacher explaining "how", "where" or "why", even if the classical description of the argument will be gradually replaced by a guide for the student that will work by himself to discover the answers.

Unfortunately, a well-known effect is that even few hours of teaching generate a huge amount of material (texts, audio, video, pictures, etc.). Managing and handling large files and ramified data structures become in practice a difficult task for both users and producers.

To guarantee the maximum lifetime and scalability of the multimedia products the reusability of the teaching material and the regularity of the hypermedia are key factors. Such requirements imply a set of technical issues regarding both data sources and hypermedia description format. In order to obtain the full modularity at each layer, starting from data up to the project schema, the original multimedia sources should preserve their own identity in the output data structure. Open data formats should be preferred for both sources and generated output. This assumption achieves the full cross-platform compatibility and allows to collect, modify and reuse the teaching material at any time and with the lowest conversion cost. It gets also through the disadvantages of most proprietary file formats, which require their own run-time libraries and are often platform-dependent.

Any automated generation system for the structure of the hypermedia navigation should respect a general set of cognitive and didactic methodologies. The organization of the didactic material depends strictly from the explicit or implicit pedagogic approaches presented by the authoring tool. These methods are difficult to understand in practice, if the author doesn’t have a deep knowledge of cognitive methods; moreover, the tool’s methodology differs in various cases from the author’s own approach. This fact makes also many authoring tools unusable by those teachers, who would experience new methodologies on the existing authoring tools that often can not be personalized. In a modern approach, alternative strategies in didactic methodology require a flexible production platform, where the same information elements preserve their own visibility and can be easily combined using different lesson description models.

A language that allows a structured, high-level design of the hypermedia project will achieve this target; nevertheless, it
allows keeping the project development at a semantic level, not depending from toolkit’s internal representation schemes. Moreover, the links in the multimedia product are semantic-driven and therefore can be automatically built from such a description. If some changes are made in the source, the final product can be automatically updated or re-built. These features minimise the manual work and restrict the focus on the content instead of presentation.

End users are commonly accustomed to the learning method imposed by the traditional didactic approach, i.e., live lessons which are normally accompanied by papers, books and personal notes. Therefore, many multimedia courses strictly respect the logical organization that characterizes traditional courses and lessons. However, it is likely that in a few years this constraint will not be present anymore. The increasing diffusion of personal and mobile computing devices, such as e-books, palmtops etc., will allow computer-based training sessions to reach a larger application horizon. As consequence, this will radically transform the way of teaching and, thus, the concept of learning. Anyhow, from the author’s point of view, even if the final product will be based on innovative didactic models the didactic elements should be collected and organised at first within a traditional schema, to guarantee completeness and to help the identification of every semantic relationship.

Low production cost and skill, easiness of usage and a short hypermedia production time are key factors for a successful didactic multimedia production environment. One of the main purposes of MTEACH is to reduce the hypermedia production time and, by extension, the time-to-market of a hypermedia-based didactic project. This is particularly important on hi-tech didactic publications due to the rapid enhancement and versioning work that makes contents obsolete in a very short time. Such products are often distributed after a long editing process, involving many resources and persons, but sometimes their contents are up-to-date in a few months after the final release. An authoring tool based on a simple and module-oriented language allows the reviewers to modify or substitute at any time the only elements in the structure, which are still under development or subject to periodic enhancements. If new arguments are added or integrated inside an existing project, the inner coherence of the structure of the whole hypermedia is maintained automatically, generating the dependence relationships among the arguments at each project build.

A recognizable modularity is further essential to obtain a good reusability of the parts of a product in a fast prototyping environment; this represents also a flexible testbed on which new didactic methodologies can be experienced and applied on advanced hypermedia solutions.

To achieve all these goals, the main effort in the MTEACH project consists in defining a methodology that preserves most of the normal preparation and teaching activity of the author and that leads to a straight sequence of production tasks. The methodology is discussed in the sections III-VII, whereas the next section addresses previous works. The last section introduces the relation between MTEACH and the innovative didactic models.

II. PREVIOUS WORK: A CRITICAL SURVEY

Common authoring tools, like ToolBook [1] and Macromedia Director [2], offer powerful functions for animation controls, programming and graphics. Authors that are not highly skilled at dealing with multimedia production are able to obtain a simple result with a limited effort. Advanced functions exceed the requirements of basic end users, which are often disoriented by an enormous choice of commands and low-level instructions.

Moreover, a considerable amount of manual work is necessary to collect the various features, especially because these tools are designed for the largest number of authors, including those for which the artistic aspect is the main part of the product. However this fact could not be a constraint in professional multimedia publishing, where authors are experienced programmers and work often in concurrent engineering teams, it often represents an unpleasant overload in case of simpler approaches for less skilled users. Another consequence is that multimedia production costs are often a big problem for high level didactic, such as for university or professional courses that cannot count on the profits of massive sells.

Further, reusability of parts of the final product is limited and additional manual work is necessary to restore all the links, in order to preserve the consistency of the multimedia data structure. This limitation descends from the strict embedding of the original data sources (audio, video and image files) inside the code generated by the authoring tool.

Cross-portability between different production environments, for example merging ToolBook and Director files, is nearly impossible, since each tool uses an internal coding system for the project description and generates archives with different formats. Working at a higher level would instead require a tool that describes the structure of the hypermedia, instead of directly implementing it.

Web-oriented authoring tools, like Microsoft FrontPage or Macromedia Dreamweaver, provide some functions to preserve link consistence when the Web content is modified, or when a part of it is moved in a different context. However, considerable work is still required to create links, to fill template pages with the available material, and to check the logical correctness and consistence of the links in the final product.

Nevertheless, the total freedom left to the author when creating the outline and assembling the collected material can lead to a hypermedia not homogeneous in its parts. This makes navigation difficult and integration between different products almost impossible. Many authors do not recognize these aspects as disadvantages, being more interested in the artistic aspects of the work. Anyhow, this is not generally true for didactic productions, where lucidity and uniformity of style are
between its various parts in terms of keywords (flat links) and hierarchical structure. In respect of commonly used authoring tools, this approach produces a more rigid, and therefore more regular, hypermedia structure. Navigation becomes easier this way for both authors and end users.

From the technical point of view, the system can be easily extended to several different platforms and operating systems, only by adding new interface modules to the code generator of the compiler. In particular, if an authoring tool provides application interfaces to external high-level programs, it can be controlled by MTEACH to reduce or eliminate the need for manual work.

Source code can be directly written and managed by the author in the MTEACH language, notwithstanding an automated code generation could be easily integrated in a graphical RAD environment.

V. ELEMENTS OF THE MTEACH LANGUAGE

The formal description of the MTEACH language, regarding the detailed explanation of syntax, lexical conventions and data types, would go beyond the purpose of this presentation paper. Nevertheless, current implementation is still under development, but its natural flexibility lets us foresee that application fields are getting larger, and new elements are to be added, according to the experimental results. Thus, a formal scheme would not still offer a complete and stable rule set for every kind of hypermedia description.

```plaintext
keyword_list ::= <identifier> 
[ , <keyword_list> ]

lesson ::= LESSON <identifier>
  KEYWORDS = { [ <keyword_list> ] }
  { arguments }

argument ::= ARGUMENT <identifier>
  KEYWORDS = { [ <keyword_list> ] }
  { contents }

arguments ::= argument [ arguments ]

contents ::= clip | visual [ contents ]

clip ::= CLIP <identifier>
  KEYWORDS = { [ <keyword_list> ] }
  MOVIE = <filename>
  { visual }

visual ::= slide | text | URL | ; visual |

slide ::= SLIDE <identifier>
  KEYWORDS = { [ <keyword_list> ] }
  PICTURE = <filename>
  TIMING = <time>
```

Fig. 1. Part of the kernel of the MTEACH language.

MTEACH provides a powerful methodology to build very large didactic multimedia products following as close as possible the normal activity of a teacher. The MTEACH language allows the teacher to describe the hierarchical structure of the lesson in the same way the index of a book is written; the keywords list associated to each element represents the logical interconnections of the arguments that
will become glossary entries and hyper textual links.

Traditional chapters become "lessons" and the sections within each chapter are called "arguments". The analogy with the book simplifies the description of the logical schema, which can be written with little effort directly from the teacher, not necessarily from a multimedia-authoring expert.

Each argument is a module that groups a collection of different types of documents together, as shown in Fig. 1. "Clips", the most important type, are made of audio/video tracks and several synchronized "visual" elements, typically "slides". Synchronization is not obtained during the audio/video editing sessions; instead, it is featured by the automatic code generation. The way synchronization is achieved depends on the target operating environment; in current implementation, for example, the Real Media [3] synchronization capability is used.

The MTEACH language requires that timing information is associated to each visual element related to the starting point of the audio/video track. A textual description of timing information could appear more complex and less effective than a graphical interface, but plain ASCII format still allow easy editing and manipulation when the description regards a large tree structure. Interactive graphical interfaces do not scale equally well, even though an additional program (with its own graphical interface) could be used to compute and insert timestamps into the source file (see section VII).

The lesson could be enriched by other documents, such as texts, animations and images even if not linked to any audio/video track. The language is currently being extended to include more data types.

A key feature of MTEACH is the automated generation of the several links, which are needed to logically interconnect topics. A list of keywords (such as "Ulnar", "Exposure", etc. in the example) is associated to each element in the description: this additional information will be used to build a multi-linked glossary during the code generation. Each keyword (it can be a word or a phrase) in the glossary will show the lists of related topics in the hierarchical index, i.e., the whole set of arguments in which the keyword is recalled. Another set of automatically generated links provides a sequential visit of clips of the multimedia lessons. Future extensions of the of the compiler will generate links for different navigation criteria.

VI. THE MTEACH COMPILER

Automatic link generation makes MTEACH different from the conventional authoring tools and provides reusability of the multimedia material after it is organized in the lesson structure. Moving a part of a lesson description (clips, slides, timing, keywords, etc.) to another project does not lose the structural information. All the links are rebuilt when the new code is generated: this is a nice feature, since even very short lessons may contain hundreds of links.

The MTEACH compiler has been developed using lex and yacc [4]. A data structure is built to internally describe the hypermedia structure and is visited during the code generation phase.

In this phase, the compiler reads a set of template files and inserts into them the actual information described by the internal data structure. This approach allows the use of conventional editing and authoring tools to build and test the templates. Of course, this is possible only if the internal structure of the file of the chosen authoring system is known, or at least some APIs (Application Programming Interfaces) are accessible to control the editing feature of the authoring tool.

Fig. 2. An example of a lesson described in the MTEACH language.

Fig. 3. A shot of the final product in a microneurosurgery teaching application.
VII. MTEACH AND REAL-TIME HYPERMEDIA PRODUCTION

The MTEACH research project is also developing a stable workspace for almost real-time multimedia production. MTEACH becomes this way the core system in the realization of a stable, real-time production environment: Fig. 4 shows the logical schema of the current prototype.

The MTEACH processing and production unit is directly connected with a digital audio and video capture system; live recording is performed through a digital video camera and a software for RealMedia or MPEG audio/video encoding. Clips can be recorded in a dedicated room or in a real classroom during an almost normal teaching session.

Firstly, the teacher prepares slides, pictures and animations to be used during the lesson. In particular, the MTEACH platform requires a preliminary conversion of the slides in the GIF standard, eventually incorporating the animations with their related timestamps. This solution has the advantage that the teacher can use his preferred graphic editor (Microsoft Powerpoint, LaTeX plus Xfig, etc.), and the conversion to the GIF format is a standard feature on most design and presentation tools. Most of the didactic material and the lesson’s schedule should be ready before the lesson starts; anyhow, the teacher can hand-draw pictures during the class, if exercises or detailed clarifications are needed.

A GIF image browser runs on the teacher’s console, which is linked to the processing unit and, eventually, with the video projector in the classroom. The interface between the console and the processing unit sends a synchronization signal in correspondence of a teacher’s action (i.e., a new slide command or a timestamp inside an animated GIF). The processing unit intercepts the signalling and includes its command or a timestamp inside an animated GIF). The correspondence of a teacher’s action (i.e., a new slide and the processing unit sends a synchronization signal in the projector in the classroom. The interface between the console is linked to the processing unit and, eventually, with the video exercises or detailed clarifications are needed.

Downloadable formats allow to take advantage of the features provided by the most common Web browsers, instead of using “ad hoc” client solutions, and can radically simplify the distribution and fruition phase of an hypermedia-based distance learning project.

The main purpose of this approach is to improve the flexibility of MTEACH and the conformance of the process output to largely used standard formats, offering it as core engine for Web-based distance learning instruments and integrated systems. Open formats allow to take advantage of the features provided by the most common Web browsers, instead of using “ad hoc” client solutions, and can radically simplify the distribution and fruition phase of an hypermedia-based distance learning project.

VIII. MTEACH AND DIDACTIC MODELS

The modularity of the MTEACH language and the scalability of its framework can be applied to various indexing methods and teaching experiences. Starting from a well-defined methodology, MTEACH allows to collect various elements of a structured lesson and assemble them in parallel learning paths; this way, a single argument or, by extension, an entire hypermedia product could be targeted to the different skill level of the audience. In fact, the MTEACH platform is going to be a modular system, where each lesson argument corresponds to a particular function, which is linked to different schemas of exposition. In other words, a certain argument could be explained in different ways, using the same set of multimedia material that refers to that particular task. In a two-dimension space, single arguments can be expanded in various description models (e.g. basic – intermediate –
advanced, or theory – exercises - practice); the result is a context-based hypermedia where the content is dynamically linked to a certain description model on behalf of the student’s skill or interest.

On the other hand, a common set of information elements can also be assembled in different versions at a certain level of difficulty, according to the different project and teaching methodologies explored by the teacher.

This approach represents a realistic substrate for dynamic hypermedia navigation, and for the integration between multimedia production and advanced systems for hypermedia navigation. Additional indexing schemes based on user models could be added to the standard method, offering a direct interface to expert systems for adaptive and user-driven hypermedia navigation, which are embedded in many Web publishing platforms. This result will be achieved by MTEACH featuring a dynamic generation of the hypermedia structure, on the basis of user-driven adaptation models; this enhancement requires an advanced indexing method, in order to guarantee the inner coherency and unity of the dynamically generated hypermedia.

REFERENCES

ABOUT THE AUTHORS

Pier Luca Montessoro received the Dr. Eng. degree cum laude in Electronic Engineering from the Politecnico di Torino, Italy, in 1986. He has been with the Italian National Council for Scientific Research (C.N.R.) from 1988 to 1992. He has also been consultant for the Digital Equipment Corporation (now Compaq) in Hudson, Mass. (USA) in the field of simulation for VLSI design. His teaching activities began at Politecnico di Torino where he taught computer architectures, computer networks and databases in a Management Engineering course from 1990 to 1994. Since November 1992 he is associate professor in computer science at the University of Udine. Author of many research papers, his research interests are actually telecommunications and computer networks, teleteaching and multimedia systems. Together with Silvano Gai and Pietro Nicoletti, Pier Luca Montessoro wrote a popular book (“Reti locali: dal cablaggio all’internetworking”, SSGRR, in Italian) on structured cabling systems, LAN and internetworking. Pier Luca Montessoro is director of the Computer Center of the University of Udine.

Davide Pierattoni received the Dr. Eng. degree in Management Engineering from the University of Udine, Italy, in 1999. Under the guide of Pier Luca Montessoro, he actually participates as contract researcher in the project on the design of new LAN architectures, at the Dep. of Electrical, Management and Mechanical Engineering of the University of Udine. His research interests are network simulation environments, routing protocols, distance learning and multimedia teaching systems. Various lab and teaching activities regarding computer networks involve him as internal collaborator for degree-level courses and for professional seminars.

Elio Toppano received the Dr. Eng. degree in Electronic Engineering (Telecommunication) from the University of Trieste, Italy, in 1981. Since 1984 he has been Assistant Professor in the Department of Computer Science, University of Udine, Italy. His research interests are in the area of artificial intelligence, and include modeling of physical systems and model-based reasoning, applications of knowledge-based systems in industrial diagnosis and design and adaptive and intelligent web-based education systems (W-AIES). Dr. Toppano is a member of AAAI, AIIA and AIED Society.
Hypermedia is not a silver bullet for decoupling client from server. The client still needs to know the structure of the resources it loads from the server and posts back to the server. This structure contains a large part of the domain knowledge, so the decoupling is far from complete. How Should I Implement My New Shiny API? In my opinion, there’s no golden way of creating a REST API. If, in your project, the advantages of hypermedia outweigh the disadvantages - mainly effort in careful design and implementation - then go for hypermedia and be one of the few who can claim to have built a glorious level 3 REST API :). If you’re not sure, go for level 2. Especially if the client is not under your control, this may be the wiser choice and save some implementation effort. MTeach is based on a methodology supported by a language and a compiler, and its goal is to help the author of a didactic production to work at a higher level than conventional authoring tools. Starting from a formal description of the lessons in MTLDL language, the running code of the multimedia application is generated in standard formats, providing a self-consistent and easy-to-navigate hypermedia structure. Citation. Montessoro, P.L., Pierattoni, D. & Cicuttini, R. (2003). MTeach: A Simple Production Framework for Context-Based Educational Hypermedia. Journal of Educational Multimedia. This article provides historical background on hypertext/hypermedia; focuses on their use in the humanities; describes humanities projects that illustrate trends and techniques; discusses libraries’ roles in hypertext/hypermedia humanities computing; and concludes with a description of challenges and opportunities as librarians implement such systems. Erwin K. Welsch, Memorial Library, University of Wisconsin, Madison, WI 55706 LIBRARY TRENDS, Vol. 40, No. 4, Spring 1992, pp. 614-46 © 1992 The Board of Trustees, University of Illinois. Welsch/hypertext, hypermedia, & humanities 615. of a shared pool of information that can be accessed nonlinearly, and Nelson was a prominent and fervent advocate of hypertext for several decades before it achieved wider popularity. Context I’m working on an intranet application for a 100000+ employees company and. Read more. good first issue. hAPI is a hypermedia web API behavioral profile and framework for creating dynamic polyglot hypermedia restful services. java hypermedia hypermedia-api. Updated Apr 9, 2018. simple CRUD application showing how HATEOAS works in a typical client/server architecture. Spring Boot Server and simple Web Client. pagination spring-boot rest-api hateoas hypermedia-api. Updated Mar 28, 2018.