A Component-based Model for Interactive Tests

Nathalie Poerwantoro[1], Abdulmotaleb El-Saddik[2], Thomas Dammann[1], Stephan Fischer[2], Bernd Krämer[1], Ralf Steinmetz[2]

1

Fern Universität Hagen
LG. Datenverarbeitungstechnik
Philipp-Reis-Gebäude
Feithstrasse 140
D-58084 Hagen - Germany

2

Industrial Process and System Communications
Dept. of Electrical Eng. & Information Technology
Darmstadt University of Technology
Merckstr. 25 • D-64283 Darmstadt • Germany

{ Thomas.Dammann, , Bernd.Kraemer, Nathalie.Poerwantoro }@FernUni-Hagen.de

{ A.El-Saddik, Stephan.Fischer, Ralf.Steinmetz }@kom.tu-darmstadt.de

ABSTRACT

In this paper we concentrate on the description of an effective way to create interactive tests. Unlike other work, our model employs a multilayered presentation to improve the effectiveness of learning. Our approach separates questions from Java applets and use the applets in conjunction with a simple authoring tool for adding questions and corresponding hints. The applets should show an animation or a simulation of a certain topic, and the question should be asked in relation not only to an applet but can also relate to other media such as video, picture, sound. We chose to develop an XML based exercise environment

1. Introduction

In the past a lot of electronic teaching material including electronic books, intelligent tutorial systems and web-based courses emerged. Most of these also commercially available products use a variety of different media such as video, audio, images, animations and hypertext to exploit the nature of hypermedia at its best in order to yield some sort of optimal learning success.

It is the goal of the Multibook project to create a system with an optimal support for the learner, integrating a variety of different learning styles and a Java-based Exercise Environment which can be used to test students remotely. As the research area of adaptive hypermedia systems is being explored for a long time now it is the particular goal of the Multibook to examine how *multimedia* can be used best to support the learning process thus extending the focus from systems most often based exclusively on text and images to real multimedia systems integrating text, images, audio, video, and animations. This also implies the knowledge which of these media has to be used with respect to the particular content to be explained. Especially the meaningful use of different media is being neglected in many commercial systems following the paradigm “the more – the better”.

The Multibook-System strives for an efficient learner-support in the process of teaching multimedia and network technology. The system consists of an electronic textbook explaining the topics to be learned. The book is extended by a didactic component which orders the content with regard to specific learning strategies, based on a user profile which guides the user by tracking the actions he is performing when reading the electronic book. An Exercise environment is also a main part of this book. The individual chapters of the book are generated on the fly on the basis of the learning style which has been chosen. We currently support hierarchical and constructivistic learning as well as learning based on examples. Hierarchical learning explains the theory of a topic followed by examples and applications. Constructivistic learning starts with applications followed by the theoretical background. Learning by example first presents examples which are then generalized to explain theories in a comprehensive manner. Using an explorative style (learning by example), a user first has to learn a specific subject by using an applet which is not of a static nature in a sense that parameters can be modified and components can be exchanged.

One of the aspects we face today is the use of applets in traditional hypertext systems as well as in Intelligent Tutoring Systems. Java applets are employed very often to explain complex processes to deepen the understanding of difficult parts of educational documents. However, the use of Java applets does not have to be restricted to simulations or visualizations of specific problems. Many examples [Han98] show that applets can be used for exercises and tests and therefore offer more flexibility than common media. Another advantage of applets is the possibility to use parameters so that they can be used for different learners and preferences. Considering the use of applets in Intelligent Tutoring System or a usual Hypertext System, they can be as intelligent as possible, but still require different pieces of information units, if they want to present the same information to users with different preferences using an applet. If it is possible to find common learning oriented descriptions for different uses of the applets and to make the applet itself adaptable to specific needs, only one version of the applet would be necessary for many different kinds of users. However, the standardization of meta data descriptions for applets is not enough to develop applets which can be used for different learning purposes. Many applets deal with very specific problems or limited topics. Our own experiences with the use of applets in the multimedia course at both FernUniversitaet-Hagen and Darmstadt university of Technology have shown, that it is difficult to use existing applets, without any changes.

This paper is structured as follows: In Section 2 we interact with aspects we identified. Section 3 examines some case studies in developing test applets. Section 4 explains the architecture of our generic exercise environment. Section 5 concludes the paper and gives an outlook.

2. Offering support through interaction

Before presenting some exercise applets and afterwards the architecture of the generic Exercise environment, the requirements for such an environment must be identified: namely *interaction* and *support* of the learner. *Interaction* implies that the learner is guided in the sense that he can get feedback if problems emerge. Assuming that the handling of the environment itself is intuitive such problems can only result from the difficulty of the topics to be learned. The *difficulty* of an algorithm to be animated can result either from the knowledge of the learner which might not be sufficient to understand the topic or from the amount of information presented by the animation. If the user's knowledge is not sufficient to understand parts of an algorithm we offer two possibilities to create the corresponding knowledge: a user can read a short explanation of the part of the algorithm he currently executes or he can invoke the chapter of the textbook explaining the underlying theory in depth. The latter includes search functions to get a more specific way of explanation.

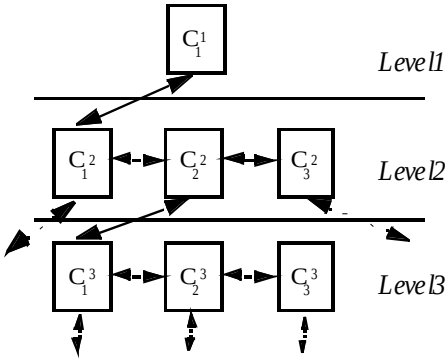

Figure1: Levels of complexity

The processing of an insufficient knowledge of a learner is performed in a traditional way by using hyperlinked multimedia documents. The second problem however, the density of the presented information has to be dealt with by another approach, the use of *levels of complexity*. The idea behind a level of complexity is that a user can reduce the information density of a part of an algorithm by splitting the part of an animation he/she is currently using into a particular number of steps which can

be understood easier equivalent to a smaller information density. This process is shown in Figure 1. While *C* stands for complexity, the upper index denotes the level, the lower the number of a component.

We distinguish between two kinds of interactions the user is provided with: content dependent interactions and content independent interactions.

Content-Dependent Interaction Model

These interactions are strongly bounded with the topic to be visualized:

- Variation of parameters of a running Applet (interactive teaching Bean)
  - Visualization (level of complexity)
  - Animation (speed, background color, foreground color,..)
  - Simulation (interactions by the user interface)

Content Independent Interaction Model

This model represents those interactions, almost all developed applications will have in common (see figures 3 and 5):

- Guiding the user
  - Help function
  - Guided tour
  - Step function
- Language (at the moment we are supporting: English, German, Spanish)
- Explanation (Errors, Hints, Audio)
- Look And Feel (Java, Windows, Motif)
- Standard video recorder interaction buttons (Start, Stop, Pause, Step-forward and step backward)

3. Case Studies

In the past, distance education has always suffered from a lack of opportunities for hands-on experimentation and access to physical objects, models, simulations and instruments in computer and engineering labs. Today, computers in education and networked learning environments offer new ways of discovery learning through some unique forms of interaction with virtual objects, online laboratories at home, learner-directed animation of realistic processes, and light-weight software tools. In networked learning environments interactivity allows students to manipulate input data, parameters and thresholds of real-time simulations or animations. Learners may also influence the progress and output of process or algorithm visualizations. The project group at FernUniversitaet Hagen is exploiting some of these possibilities with the development of a series of exercise environments that let students apply and experience what they have learned to predefined problems or problems of their choice. Thereby we expect to raise the attractiveness of the learning material and enhance the students' motivation, especially that of distance learners. In the design of these environments we have attempted to associate learning objectives with appropriate multimedia solutions with respect to different teaching functions such as motivation, presentation of new structures, practicing new behavior, modeling, exploration by trial use and error, or application of new techniques. To achieve a high degree of platform independence, ease of use and portability and to remain independent from proprietary formats, it was decided to rely on HTML, XML and Java as core technologies for implementation of the exercise labs. In the sequel, we shall sketch the design and functionality of interactive learning environments supporting home practices on two different topics of the course content: CORBA and SGML.

The CORBA interactive learning environment consists of three parts : theoretical aspects of CORBA, distributed application development with CORBA, and “CORBA in action”. The first part consists of a hypertext document explaining all CORBA components in detail. The root of this hypertext document starts in a visual model of the standard Object Management Architecture (OMA) shown in figure 1. The second part leads the student through different steps of a CORBA-based application development (figure 2). The third part consists of a learner controlled animation visualizing the various steps such as transformations, looking for and connecting to a server application, requesting and answering a request, that take place underneath the application surface where remote objects interact with each other. Students can navigate in the learning environment by selecting a topic depicted in an image. The navigation in the lessons is not based on a textual table of contents but is based on an applet showing an image which acts like a toggle button.

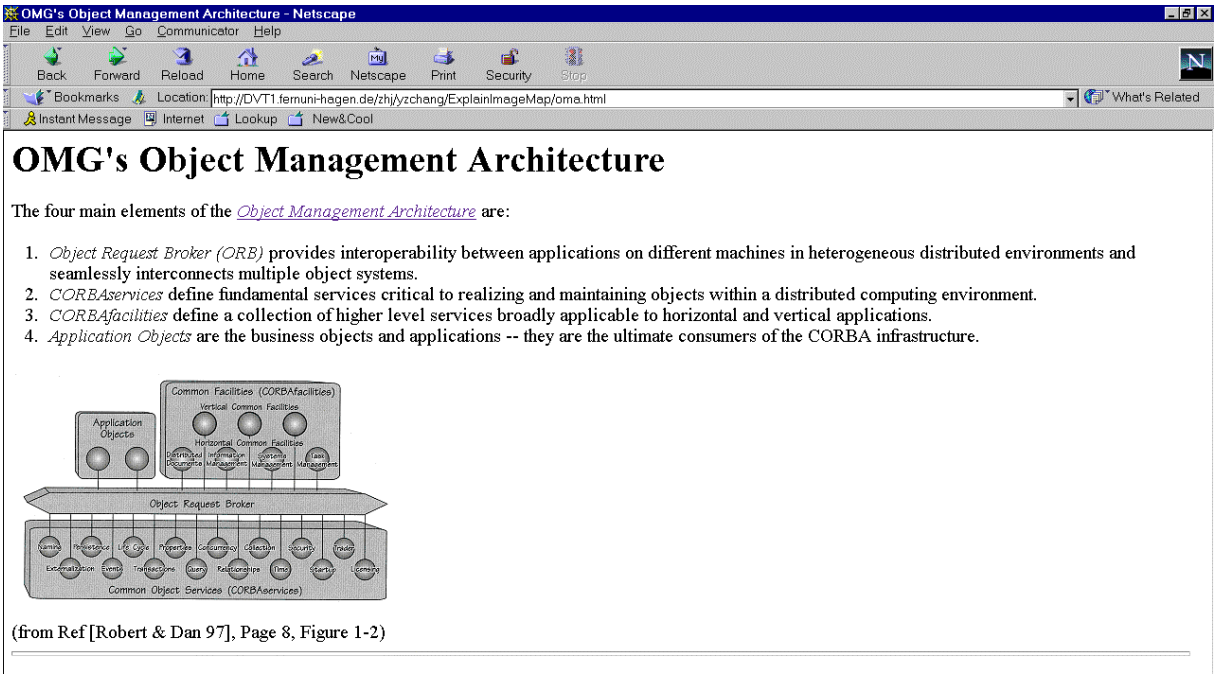

Figure 1: Table of contents map applet for CORBA tutorial

After completing the theoretical part of the learning environment students can try an applet which explains the topics of CORBA by using examples.

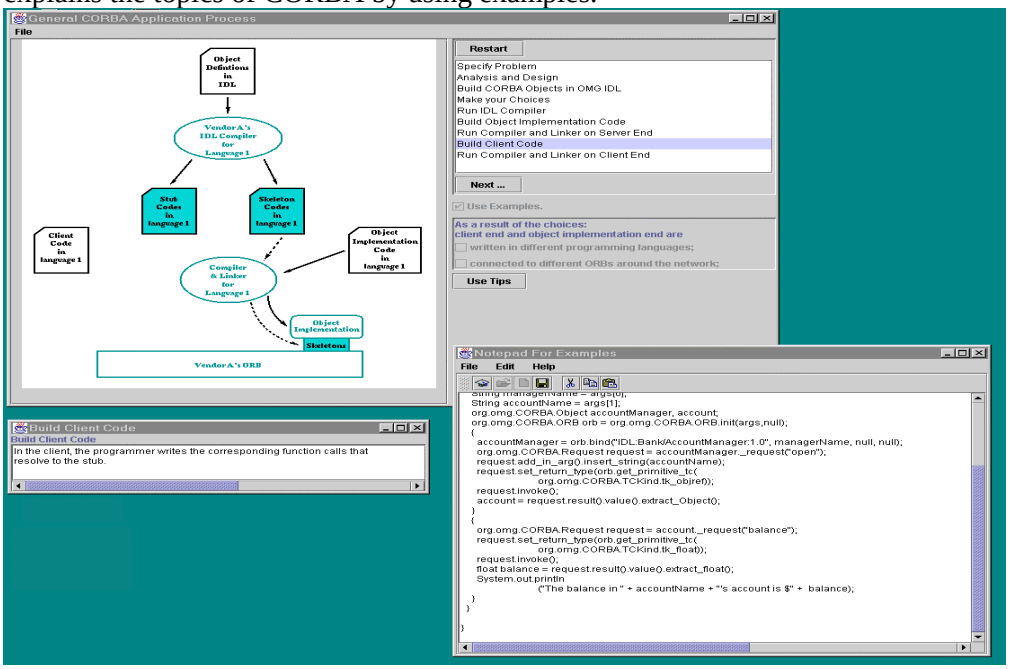

Figure 2: Applet simulating the steps to create a CORBA application

The steps in creating a CORBA application are illustrated with various possibilities. Students are guided through the CORBA application creation process one step at a time. A notepad is provided where students can view the IDL file and source codes of the examples (bottom right frame in figure 2). A detailed explanation to each step can be viewed on a separate frame (bottom left frame in figure 2). This application also allows students to develop their own example by writing their own IDL files and source codes and sending it to the university server to be compiled. This feature itself is implemented through CORBA. A CORBA server application at FernUniversität services requests submitted through button clicks in the 'General CORBA Application Process' section to compile the IDL files or code and return a list of file names generated.

CORBA exercise applets

Once the students have deepened their insight into the CORBA middle-ware, they have to solve several problems posed along on online bookshop example (figure 3).
A simple online bookshop is used to asses a student's knowledge of CORBA. The bookshop itself acts as the service provider and sits on the server side. In the example, the bookshop has to service requests coming from the client using static invocation and dynamic invocation. The naming service is also included in one exercise problem. Based on the bookshop example, 5 types of exercise applets were created, which are grouped into 2 difficulty levels: Static Server, Portable Server, Static Client, Client using Dynamic Invocation Interface (DII), Naming Service. The goal in the design of these exercises is to show the steps to be taken in the creation of a CORBA client or server and not only the source code, which shows how a client or a server would look like if programmed in JAVA. By showing these steps in pictures, the students have a better insight and understanding of the concept of CORBA, which is a more important aspect in learning this technology than writing codes. The exercise type in this topic is positioning the answers into the right order in the sequence. Each answer consists of a step in the sequence and several lines of code that are connected to this particular step. The code lines are inserted into the source code in the position selected by the students. The feedback to the students consists of results, hints and complete answers to all questions.

The students can submit the answers any time in the exercise and get the results and hints as a reaction. The hints are only given in the case of wrong answers. Should the student give up, the complete answer can be requested from the applet by choosing the surrender button. In case of errors, the students can try to re-answer the question by undoing the entered answer and submitting a new answer.

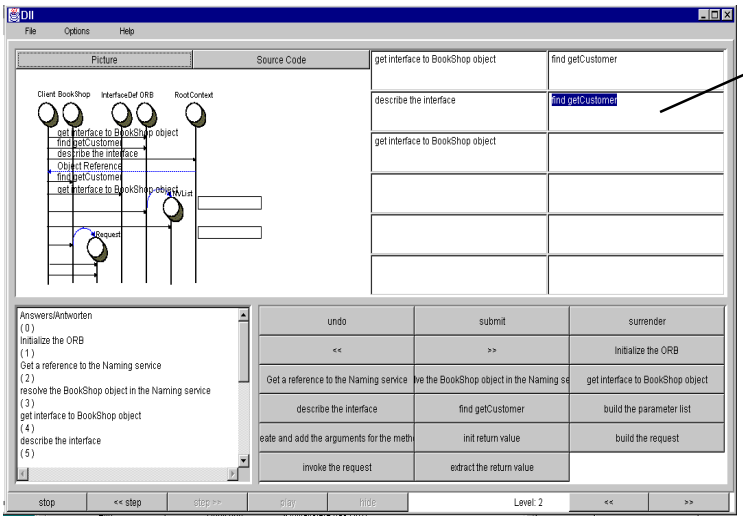

Figure 3: exercise applet

The current position is highlighted in the answer section

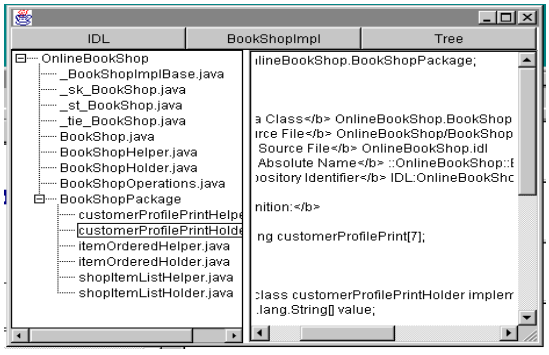

Figure 4: IDL files and source codes as exercise companion

A companion to all exercises is provided, too. This frame shows the IDL file of the bookshop(figure 4), the source code of the bookshop implementation and a tree showing all the files generated by the IDL compiler.

Exercise feedback 'bookshop in action'

The students can run the bookshop and see what the exercise they had finished earlier is all about (figure 5). Two types of bookshop servers were created, one with and another without naming service and three clients, which show static invocation, naming service and dynamic invocation interface.
The client applets design:
All the clients are designed to show the steps that occur while the clients are :

- Connecting to the server
- Requesting a service from the server

The basic steps showed by each client at the beginning are the process of connecting to the bookshop server and requesting the list of available books. All other steps that follow these opening steps depend on the actions of the students using the client.
Features of this book shop example are:

- A new customer can register himself by entering some personal data
- Registered customer can request his/her profile by entering his/her ID
- The customer status will be updated each time a transaction occurs (the customer buys some books)

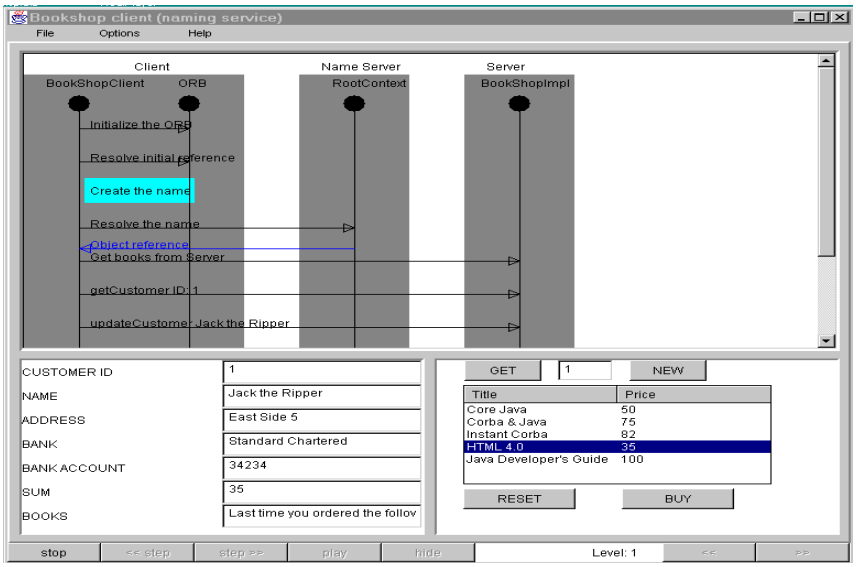

Figure 5: The diagram is updated on each interaction between the client and the server

SGML Exercise Applet

We also provide an applet dealing with the Standard Generalized Markup Language, SGML, because of its universal importance to multimedia and electronic publishing. SGML is a meta language that is widely employed in the publishing sector and it is used to define HTML. It is the first applet we developed in the context of this project.

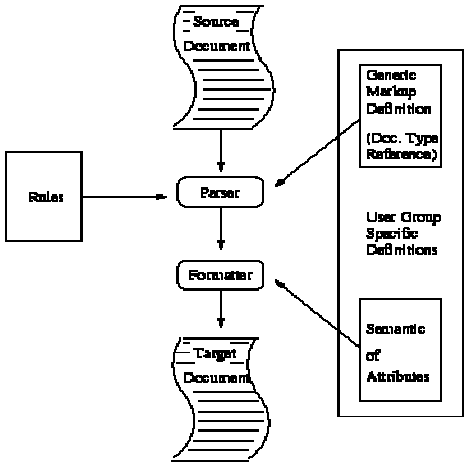

Figure 6: SGML document processing: from information to presentation

A parser-based exercise applet was created to deal with SGML topic. The parsers needed in these exercises are generated with the help of JavaCC tool, a Java compiler-compiler from Sun Microsystems. This tool allows one to generate compilers from an LL(1)-grammar of the language considered mixed with Java code. The tool is freely available. Students must answer questions concerning creation of DTDs by entering the needed tokens on the left text area. The GUI frame is composed of two main areas: The input area on the left hand, and the output area on the right hand (see Figure 7).

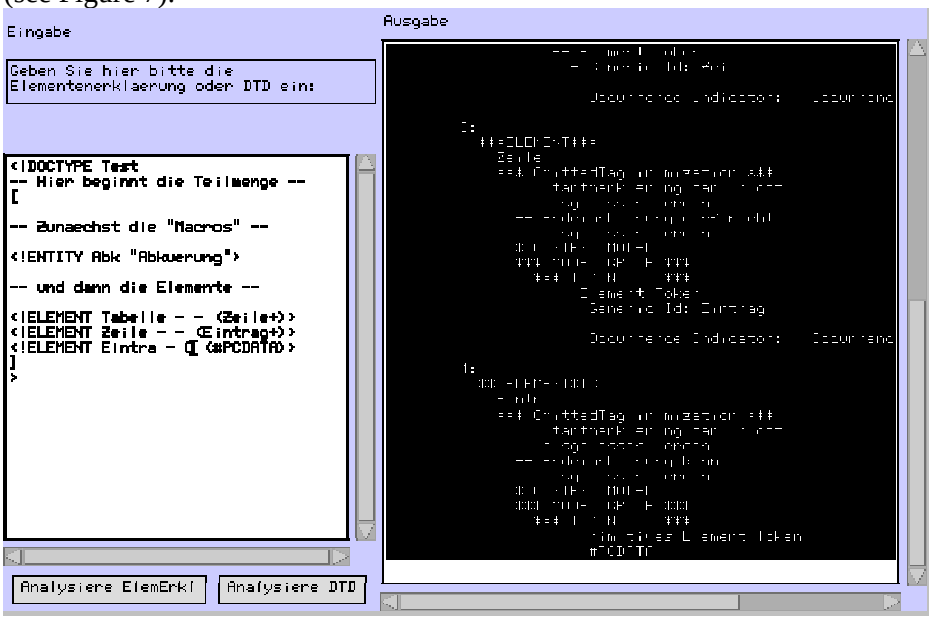

Figure 7: A screenshot of the SGML exercise applet

All text fields will be referred to symbolically and all text variables will be collected in a single class to simplify the adaptation to other languages. For each exercise, a separate parser is available to analyze the student's input for the actual exercise.
The task of the parser is to:

- analyze the input and check its syntax and
- generate a data structure representing the parse tree.

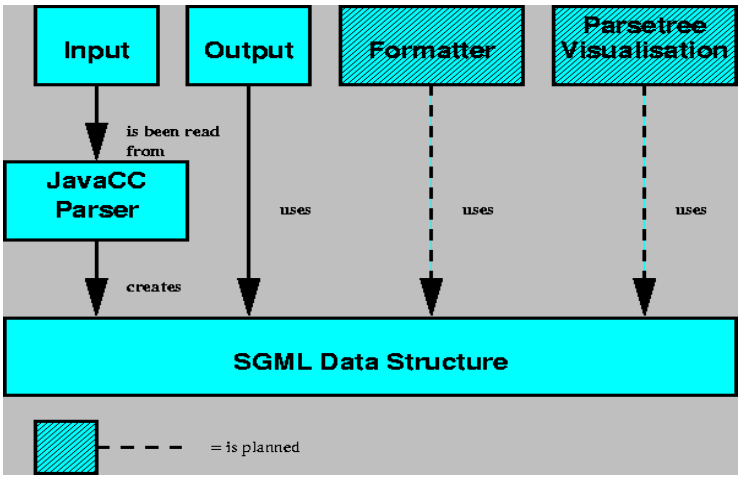

Figure 8: The programming structure of the SGML exercises

The following SGML exercises exist: comment declarations, SGML names, the structure of the document type declaration (DTD), the entity declaration and the DTD including such declarations, the element declaration and the DTD that may contain entity and element declarations.

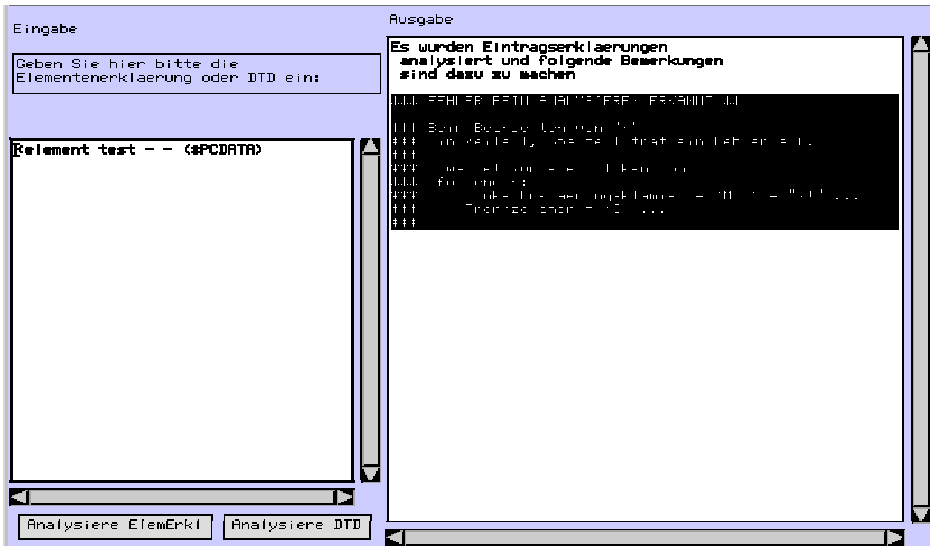

Figure 9: An example of the element declaration exercise applet with errors in the input pane

In Figure 9, we show how a faulty input looks like : an error description appears in the output area, as shown in Figure 9. This error message indicates the position where the error occurs, here at line 1, column 1. Then the expected tokens are listed.

4. A Generic Exercise Environment Creator

After having developed several exercises, we realized that we need much more exercises for the students to make the self assessment effective. Developing an exercise applet for each topic by integrating the questions in the applets by means of cut and paste technology was not very effective. A better approach would be to separate the questions from the applets and use the applets in conjunction with a simple authoring tool for adding questions and corresponding hints. The applets should show an animation or a simulation of a certain topic, and the question should be asked in relation not only to an applet but can also relate to other media such as video, picture, sound. We chose to develop an XML based exercise environment. XML is used to describe the question and the exercise as a whole. The idea was to describe each question in XML and let those question be parsed during the exercise. That way a big amount of question pool can be generated more easily. The configuration of an exercise is also described in XML and parsed at the beginning of an exercise. The exercise environment covers both formative and summative assessment types.
The exercise environment consists of 3 modules or applications:

- **Question generator**
  A Java application with forms to create a question. Five of the most common question types in Computer Based Training (CBT) can be generated such as: true/false, multiple choice, fill text, matching and set ordering. An XML Document Type Definition (DTD) was defined to describe the structure of questions. The IEEE standard for Learning Object Metadata (LOM) was taken as guide to define the set of relevant fields to describe a question.
  Important parts in a question description include :
  - Domain: defines to which domain or topic the question belongs
  - Create: defines the date of creation
  - Level: defines the difficulty level of the question. 6 levels are currently supported
  - Language: defines the language for the question
  - Subtopic: defines the subtopic or subchapter of a question.
  - Question type: defines the question type which has to be one of the 5 described above
  - Question: defines the elements that describe a question such as
    - Problem statement : defines the question asked
    - Answer options : defines the answer options in question types multiple choice, matching
    - Keys: defines the answers to the question
    - Hints: defines the hints, each hint is related to an answer option and the hints consist of 3 levels, starting from a general hint to a detailed hint
    - Reason: defines the 'why' explanation of the answer
    - Show media: defines the media related to the question. The media can be a picture, a video clip, audio , applet.
  - Picture list: defines a list of picture needed for the question type matching and set ordering

- **Exercise generator** (figure 10)
  A Java application for creating the overall exercise specification. The exercise is described using the following fields:
  - Domain: defines the topic of exercise
  - Level: defines the difficulty level of the exercise
  - Number of sections: defines the number of sections in the exercise, 5 is the maximum since there are only 5 question types available
  - Section Description: specifies each section by defining:
    - Section type: defines the question type of this section
    - Number of question : defines the number of questions in this section
    - Total question: defines the number of available questions in the question pool If the number of questions needed in an exercise section is smaller than the available questions in the question pool, the exercise tool will select the questions randomly from the question pool. The exercise is less predictable and becomes more

challenging since the students do not get the same questions each time they start the exercise.

- Description of test mode:
  - Maximum error in relation to minimum solved questions. These 2 numbers are needed to define the error rate allowed in a test
  - Time limit can be defined in test mode

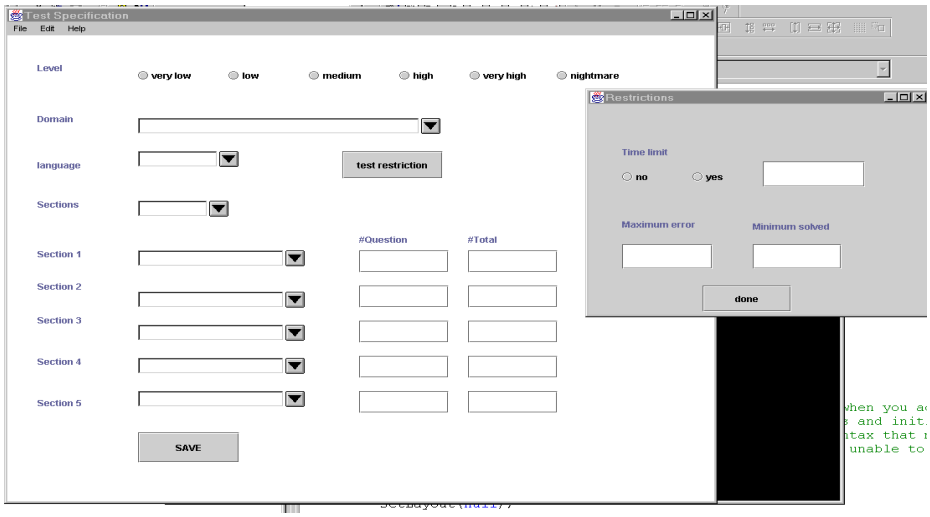

Figure 10: Exercise generator

- **Exercise application** (figure 11)
  A Java applet which is the environment that the students will work with. Students can select the desired exercise topic, language, difficulty level and mode. The applet then loads the selected exercise scenario and parses it using IBM XML parser. Using the extracted information about the exercise, the applet prepares the exercise and loads the defined questions one after another. Each question is loaded, parsed and shown when the student arrives at the specific question. If a certain media is related to the question such as an applet showing an animation or a simulation, a video clip, an audio file or a picture then it is shown together with the question. Questions can either be answered or skipped. After arriving at the last question the student can submit the exercise and get the result of the exercise. Hints are also provided and can be called during the exercise. Each possible answer option has 3 levels of hints starting from a general one to a more detailed one. The number of hints that can be shown to the student depends on the difficulty level of the exercise. No hints are available in test mode. In test mode a time limit can be activated which limits the time given to the student to finish the test. When the time is up the test will automatically end and no further answers will be accepted

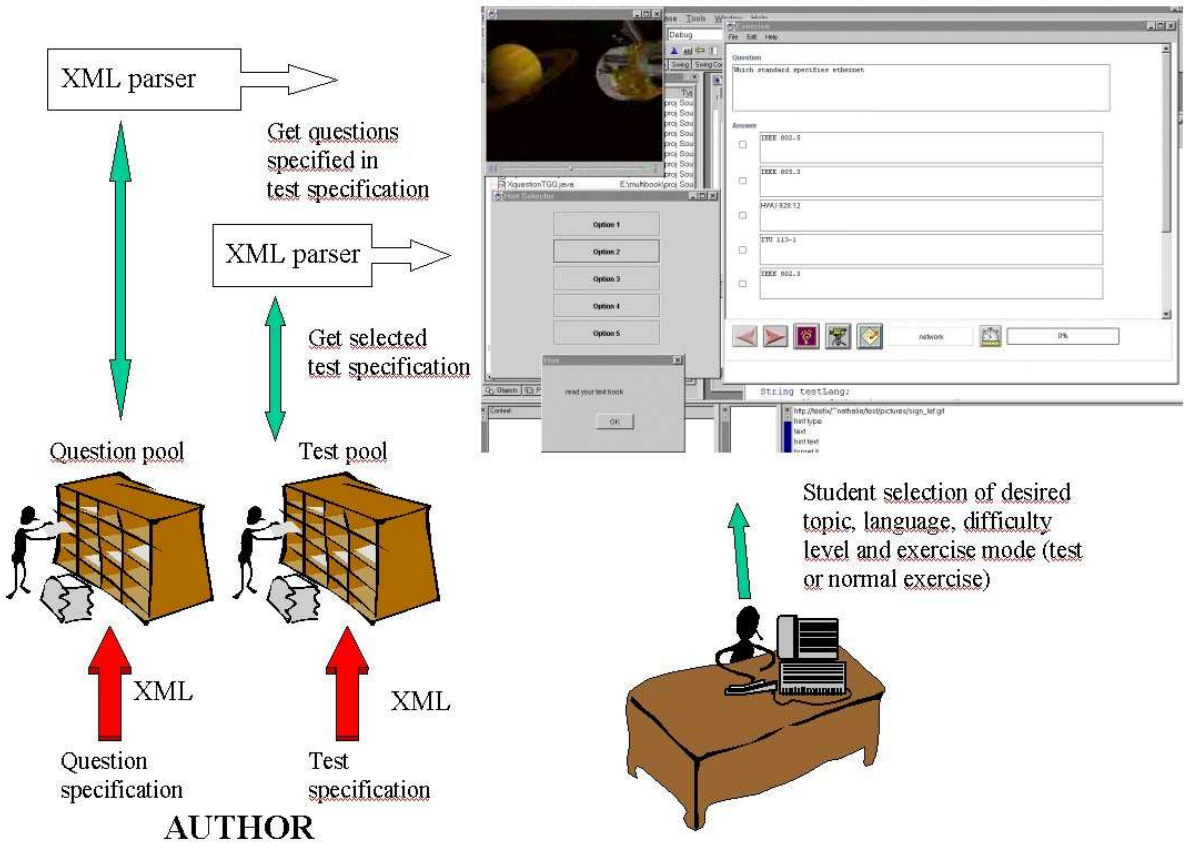

Figure 11: Exercise environment

5. Conclusion and outlook

In this paper we described a way to create interactive tests. Our approach extends other concepts by the use of hierarchies thus supporting the learner in an efficient way. We developed an interaction model in order to let the user concentrate on the topic to be tested instead of looking for the right buttons to achieve certain task. At the moment we study how the exercise environment can be integrated into the multimedia technology lessons for student self assessment.

References:

Andreas Vogel, 1997
Andreas Vogel ; Keith Duddy Java programming with CORBA , Wiley, 1997, New York

Cay S. Horstmann, 1998
Cay S. Horstmann ; Gary Cornell Core Java 1.1 Part I , Sun Microsystems Press, Mountain View, Calif.

Cay S. Horstmann, 1998
Cay S. Horstmann ; Gary Cornell Core Java 1.1 Part II, Sun Microsystems Press, Mountain View, Calif.

Charles F. Goldfarb, 1990
Charles F. Goldfarb. The SGML Handbook. Oxford University Press, New York, first edition, 1990.

Elizabeth J. Gibson, 1996
Elizabeth J. Gibson, Patrick W. Brewer, Ajay Dholakia, Mladen A. Vouk, Donald L. Blitzer. A comparative analysis of web-based testing and evaluation systems, Dept. Of Computer Science North Carolina State University, Raleigh, NC, 1996

Elliotte Rusty Harold, 1998
Elliotte Rusty Harold. XML Extensible Markup Language, IDG Books Worldwide, 1998, Foster City, CA

Frank Farance, 1998
Frank Farance, Joshua Tonkel. Learning Technology Systems Architecture Specification, Farance Inc., 1998

Gary Williams, 1997
Gary Williams, Liz Smith, Tom Lawson, Bill Lord, Deri Hadler, Roger Murphy, John Merkel. Choosing the type of assessment, Center for Enhancing Learning and Teaching (CELT).

Gary Williams, 1997
Gary Williams, Liz Smith, Tom Lawson, Bill Lord, Deri Hadler, Roger Murphy, John Merkel. A guide to the educational use of multimedia, Center for Enhancing Learning and Teaching (CELT).

Hansen, S. R., Schrimpsher, D. & Narayanan, N. H. (1999).
From algorithm animations to animation-embedded hypermedia visualizations. Paper accepted to the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA'99), Association for the Advancement of Computing in Education.

Helmut Herold, 1995
Helmut Herold, lexikalische und syntaktische Analyse, Addison-Wesley, Bonn, second edition, 1995

IEEE, 1998
IEEE Learning Technology Standards Committee (LTSC). Learning Object Metadata (LOM) Draft Document v2.5, Dec 1998

Joanna Bull, 1996
Joanna Bull. Computer based assessment: some issues for consideration, CTISS Publications, 1996

Nazmul Idris, 1999
Nazmul Idris, Sun Koh. XML and Java Tutorial, The Bean Factory, LLC., 1999

Nora Mogey, 1997
Nora Mogey, Helen Watt. The use of computers in the assessment of student learning, Learning Technology Dissemination Initiative, 1997

Ralf Steinmetz,1998
Ralf Steinmetz. Multimedia-Technologie, Springer-Verlag, Berlin, Heidelberg, second edition, 1998.

Robert Orfali, 1998
Robert Orfali ; Dan Harkey. Client server programming with JAVA and CORBA, Wiley, 1998, New York

Robert Orfali, 1997
Robert Orfali; Dan Harkey; Jeri Edwards. Instant CORBA, Wiley 1997, New York

Sun Microsystems Inc., 1998
Sun Microsystems Inc., The Java CC API Routines, http://www.suntest.com/JavaCC/DOC/apiroutines.html, 1998.

Sun Microsystems Inc., 1998
Sun Microsystems Inc., Description of the JavaCC Grammar File, http://www.suntest.com/JavaCC/DOC/javaccgrm.html, 1998.

Sun Microsystems Inc., 1998
Sun Microsystems Inc., JavaCC Homepage, http://www.suntest.com/JavaCC, 1998.

Sun Microsystems Inc., 1998
Sun Microsystems Inc., Java Media Framework Programmer's Guide, 1998