Visual Language-Based System for Designing and Presenting E-Learning Courses

Gennaro Costagliola, Università de Salerno, Italy
Filomena Ferrucci, Università de Salerno, Italy
Giuseppe Polese, Università de Salerno, Italy
Giuseppe Scanniello, Università de Salerno, Italy

ABSTRACT

In this chapter we present a system supporting instruction designers in the design and deployment of e-learning courses. The system includes integrated modules for several authoring activities, such as the definition of knowledge content objects, and the creation of assessment and self-assessment tests. The distinguishing characteristics of the proposed system is that it is based on a suite of visual languages, enabling the modelling of different aspects of the construction process for Web-based distance courses. The languages include a Learning Activity Diagram, which extends UML Activity Diagrams to make them suitable for modelling distance course structures; a Self-Consistent Learning Object language used to define knowledge contents; and a Test Maker Language for specifying assessment and self-assessment tests. The use of visual languages provides an intuitive and friendly system user interface that allows instruction designer to easily compose and analyze the distance course structure and keep track of the knowledge acquisition process individually for each learner.

Keywords: authoring tool; e-learning; instruction design; LMS; visual languages

INTRODUCTION

We are living a phase where learning is vital to many aspects of our life. This entails a continuous elaboration and augmentation of knowledge, which can be enhanced through educational and informative tools, and built upon advanced devices, such as desktops, laptops, tablet PCs, PDAs, TV and mobile phones. Moreover, when connected to the Internet, these devices might replace traditional classroom and distance courses. E-learning provides the technical infrastructure to let us define, support and deploy knowledge for new emerging paradigms of teaching (Schar & Kriger, 2000). Thus, traditional learning activities, such as lectures, homework and assignments, are replaced with new, powerful and fascinating teaching opportunities known as well as e-learning activities.

E-learning exploits Web technology as its basic technical infrastructure to deliver knowledge to the learners' environ-

ment. As the current trend of academic and industrial realities is to increase the use of distance learning, in the near future a higher demand of technology support is expected. In particular, software tools supporting the critical task of instruction design should provide automated support for the analysis, design, documentation, implementation and deployment of instruction via Web.

Muraida and Spector (2002) and Kasowitz (1997) review much of the work done in automated instruction design support tools. In particular, Muraida and Spector assert that there is "a lack of instructional designer expertise, pressure for increased productivity of designers, and the need to standardize products and ensure the effectiveness of product." Thus, tools supporting instruction design during all the phases of the learning process definition are desirable. Goodyear views the instruction design as falling within four main approaches (1997). These approaches allow the instruction designer to generate e-learning activities from given specifications by means of tools supporting the design of course structure, the selection of presentation templates, the reuse of design elements and the coordination of activities accomplished by a design team. Moreover, Goodyear also proposes an approach for analyzing and designing distance courses that is divided into neat parts (1999). The first part of Goodyear's approach resembles the work of other people (outside education) interested in the design of technology that supports the work of information systems designers, requirements engineers, human factors specialists, and so on. The second part is focused on the design of good learning tasks, exploiting traditional analysis and design processes. Often, these tools are not able to compensate the lack of expertise of instruction designers. Jones, Shirley and Lynch (2003) have presented an Information Systems Design Theory for the design of Information Systems to be used in Web-based Education. Vrasidas (2002) presents and discusses a system to develop hypermedia approaches as part of courses and learning environments delivered on the World Wide Web. It details the structuring of information, branching and interactivity, user interface and navigation through Web-based distance courses.

As opposed to these approaches, which are based on traditional models of instruction design, there are approaches and tools relying on object-oriented models. Douglas (2001) proposes an instruction design methodology based on the object-oriented paradigm. Designer's Edge (2003) provides another interesting approach and tool for instruction design based on the object-oriented paradigm.

Despite the fact that there exists a large number of software tools, none of them supports the component-based instruction design through the reuse of learning content objects. Many instruction design approaches proposed in the literature are based on traditional pedagogical learning approaches, or on the object-oriented paradigm. Indeed, existing models of instruction design have been influenced by linear or object-oriented software development processes. It is worth noting that both approaches can only partially support knowledge content reuse, since only one level of granularity (or at most two) is permitted when trying to reuse predefined content objects.

New trends seek means to exploit ideas and benefits of component-based approaches for implementing and delivering learning environments. In particular, the idea is to reuse learning components at different granularity levels. At the topmost level there exists self-consistent learning contents that may be composed of learn-

ing objects, which in turn may be composed of raw contents. It is worth noting that the self-consisting learning materials can be seen as a framework in which instruction designers insert learning contents and raw contents specifying their interrelations and dependencies. Therefore, it could be interesting to reuse a self-consistent learning content and possibly also the associated learning objects.

The visual language-based approach we propose in this paper supports the definition of e-learning processes assembling predefined didactic activities. The learning contents can be broken down and structured into a hierarchy from smaller, lower order blocks of material to higher, more complicated levels of learning. In particular, we have identified three different granularity levels referring to the size of knowledge contents. The use and assembling of these knowledge components provides the instruction designer with a modular paradigm to create distance courses, which resembles software development processes based on visual languages (Costagliola, Ferrucci, Polese, & Scanniello, 2003; Ferrucci, Tortora, & Vitiello, 2002; OMG Group, 1993). Hence, a hierarchy of three visual languages has been employed during the different phases of the distance courses' design process. Based on these languages, we have constructed the System for E-learning Activity MANagement (SEAMAN) to provide automated design support. The system and underlying approach are particularly suitable for learning methodologies centered on didactic materials and assessment rules. Moreover, the generated courses can be deployed in any Shareable Content Object Reference Model (SCORM) (ADL, 2003) compliant Learning Management System (LMS).

The first proposed visual language extends the Activity Diagrams of Unified Modelling Language (UML) (OMG Group 1993) to enable the specification of didactic contents, assessment activities and their relationships. For that reason, such diagrams are named Learning Activity Diagrams (LAD). They provide an explicit way to represent complex relationships between structural and behavioral e-learning activities. Although the language aims to support the design and development of learning processes, it has turned out to be a powerful tool for presenting e-learning activities and for monitoring students' progress. However, such diagrams are also useful for students who can monitor the learning activities they have accomplished. In fact, students can self monitor their progress thanks to the diagram animator provided in the SEAMAN's server module. As a consequence, LAD provides a powerful tool for learning traceability. Any activity specified in a LAD sentence can be further refined by using a visual sentence belonging to either the Self-Consistent Learning Object (SCLO) language or the Test Maker Language (TML). SCLO is a special case of state transition diagrams that enables the instruction designer to define learning content objects. In particular, the instruction designer defines and creates contents using predefined graphical layouts. Instead, TML extends state diagrams to enable the design of assessment and self-assessment tests. It lets us describe tests that adapt themselves to students' answers. It is worth noting that tests play an important role in our approach, allowing us to define learning processes adapting themselves to student performance.

This article is organized as follows: The next section presents the hierarchy of visual languages and provides examples of visual sentences for each language. Then we describe SEAMAN's system architecture, its facilities and a sample application.

Finally, a discussion on the achieved results and future work concludes the paper.

VISUAL LANGUAGES

Visual and diagrammatic representations play a central role in several application domains, since they provide important tools for describing and reasoning. As visual languages have been applied to new application domains, such as spatial databases, education, software engineering, and so forth, many different types of visual notations have been devised. These provide means to easily capture and model difficult concepts underlying the structure of a problem, and facilitate finding a possible solution. Thus, they can potentially improve productivity in different application domains. In particular, in the software engineering domain they are widely employed for supporting the phases of the development process, such as requirements specification, analysis and design (Ferrucci et al., 2002). The numerous analogies between software development and instruction design processes suggested exploiting visual languages to support several tasks of the instruction design process, such as the abstract modelling, the refinement of models, their assembling, and so forth. Thus, the three visual languages we propose for supporting instruction design extend or resemble languages that have been successfully used in software design and other application fields.

The main language is the LAD (Costagliola et al., 2003), which extends UML Activity Diagrams with means to model the workflows of learning processes (OMG Group, 1993). LAD can be used to model e-learning activities composed of distance modules, assessment and self assessment tests. States in LAD are activities, and most of the transitions are implicitly triggered upon completion of the actions associated to the learning activities. The second visual language we have defined is used to refine the learning content objects, whereas the third is employed to design the assessment and self-assessment tests. These visual languages — SCLO and TML — have been introduced to refine the basic objects used within Learning Activity Diagrams. We describe them in the following sections.

Learning Activity Diagrams

Learning Activity Diagrams are a special case of state diagrams where the states are SCLO objects, and transitions are triggered upon the completion of them. The purpose of LAD is to model workflows associated with distance educational processes. As a consequence, it allows instruction designers to describe educational materials, dependences and assessment rules. Material dependences allow the author to vary the degree of control over the order in which students must explore the materials spread in SCLO objects. Moreover, using the results of assessment or self-assessment tests, the flow of the learning process is adapted to the learner's performance. For example, before taking up a course the instruction designer can define a student test whose result may be used for assessing the knowledge, and to properly adapt the student learning process.

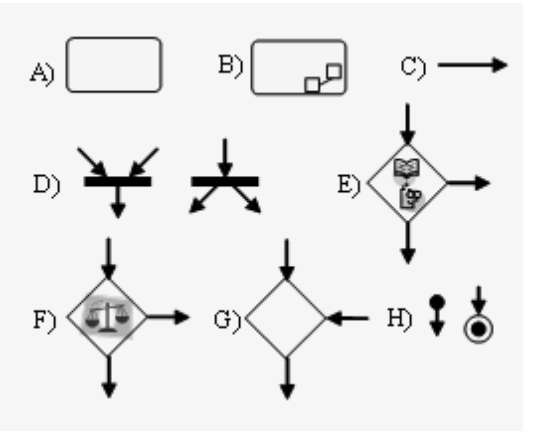
UML Activity Diagrams have been extended, to enable the modelling of the learning process based on asynchronous e-learning activities. In particular, by removing the synchronization bar elements from UML Activity Diagrams, we derive a special case of flow diagram, but with a considerably reduced language power. In fact,

we cannot describe activities without dependences; hence, all the activities have to be consumed in a sequential fashion. In the definition of processes focused on flows and driven by internal processing — for example industrial, didactic and software processes — this type of behavior is vital.

The visual language symbols are shown in Figure 1. The first symbol (Figure 1A) represents a SCLO object, or content object. The name of the Object can be placed in the symbol. This symbol represents a state of a learning process that is left when the associated learning object is completely executed. Every learning object can in turn be separately analyzed and refined by using another visual language (Figure 1B). When this happens, the icon shows a nested structure. The arrow (Figure 1C) represents the transition symbol, and it can contain a label. When the transition is not labelled, the only result of interest is content object completion. In cases where it is important to know which content object has been completed, we associate different colored coins to content objects' execution. The Synchronization symbol (Figure 1D) is a thick horizontal bar, and is used to coordinate content objects. The actions underlying content objects may be concurrently executed more than once. The number of concurrent invocations is determined at run time by a concurrency expression. The synchronization bar provides a simple way to express concepts like waiting for concurrent content objects to finish before proceeding forward along the learning process (join), and the starting of several content objects in parallel (fork).

Let us now introduce two symbols describing assessment and self-assessment activities. The assessment test symbol (Figure 1F) is used to represent an activity aiming to evaluate the learner knowledge. The self-assessment activity (Figure 1E) is meant to be accomplished by students to assess their knowledge. As a consequence, we differentiated the notations for these two symbols. Both symbols are used to represent decisions. As an alternative to guards on separate transitions leaving the same state, the aim of these objects is also to synchronize incoming activities. These

Figure 1. LAD icons: A) SCLO object; B) refined self-consistent learning content object; C) transition element; D) synchronization bar element; E)self-assessment element; F) assessment element; G) merge element; H)start and stop marker



have one or more incoming arrows and one or more outgoing arrows. The guard conditions are used to indicate different possible transitions that depend on test results. A decision may be shown by labelling multiple output transitions of an action with different guard conditions. These guard conditions may depend on self-consistent learning content objects that the instructor has to assess.

We have used a merge symbol (Figure 1G) to merge back decision branches. A merge has two or more incoming arrows and one outgoing arrow. As opposed to the synchronization bar, the incoming transitions are not synchronized.

Content experts, instruction designers, instruction technologists, media developers and evaluation specialists are professional figures that could be involved in the distance learning development process. For this reason, learning objects may be organized into swim lanes; the lanes can be used to organize learning objects with respect to these professional figures.

The last two symbols (Figure 1H) are the start and stop markers. They are used to indicate the initial and final states of a diagram.

LADs aim to provide powerful means for modelling e-learning courses. Although LADs have been conceived for courses based on asynchronous activities, we are currently extending it to enable the modelling of synchronous activities. This will require the introduction of suitable interaction paradigms and new icons in the language to appropriately support such a teaching approach.

Self-Consistent Learning Object Language

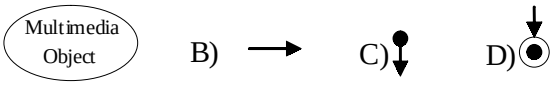
The SCLO language is a kind of state transition language. Before defining it, we have investigated several languages used in Multimedia Software Engineering (MSE). However, most MSE languages are complicated, requiring high expertise to be used. Our aim was to formalize a visual language to be easily used by the target user within the visual environment implementing it. Thus, we have first defined a graph language to describe multimedia contents of courses.

Four icons have been used to define this language. These are the multimedia node, multimedia link, and start and stop marker node. Multimedia nodes (Figure 2A) represent the educational content that will be presented to the student. Typically they are composed by one or more multimedia raw contents. Thus, this node can be an atomic element or it may be composed aggregating atomic multimedia contents. The atomic content elements can be a single Web page or simple multimedia objects, for example, short movies, songs, jokes, images, simulations, and so on. The multimedia object can have one or more incoming arrows and one or more outgoing arrows. Two multimedia nodes can be joined through a multimedia link. This represents that students can browse multimedia contents by crossing links connecting objects. The last two symbols are the start and stop markers (Figure 2C and 2D). These are used to indicate the initial and final state. The SEAMAN system we present in the paper implements this visual environment by presenting a predefined page layout to the lecturer that can be used for managing knowledge content objects.

Test Maker Language

Student assessment and self-assessment is a critical task in the knowledge process (Cynthia, Finelli, & Wicks, 2000;

Figure 2. The visual language elements: A)multimedia node or knowledge fragment; B)multimedia link; C)start marker node; D) stop marker node



Safoutin, Atman, Adams, Rutar, Kramlich, & Fridley, 2000). The literature proposes a wide range of authoring tools to construct tests. Often, these tools do not have an associated visual environment to describe the test structure and its contents. In this chapter we introduce TML, a visual language supporting teachers during the design and implementation of tests. The language provides means to describe tests that adapt their contents to student answers. TML has five different symbols and three link types. The symbols are: question, aggregation symbol, multimedia object, and start and stop marker. The three link types define transitions between language symbols. Each of them has a different color.

The instructor can use the question symbol to represent one question and its associated answers, or refine it by using symbol annotations. Annotation is performed by using visual sentences from the same language.

Question symbols (Figure 3A) are grouped with respect to knowledge contents and swim lanes. These are also used to give an execution order to regrouped answers associated to knowledge contents; the swim lanes order is from left to right.

Links are used to model answers to questions. Thus, when a language sentence has an *any link* (Figure 3D) it means that the following question does not depend on the answer. Conversely, the false and true links (Figure 3E and 3F) modify the student test structure. The last symbol (Figure 3C) is recommended to motivate answers.

Some Examples

Here we provide an LAD and show how to use it, to enable students and lecturers to monitor the learning process. Moreover, we present a sentence for each proposed language. It is worth noting that students cannot inspect sentences that are refined through LAD symbols. Thus, they can only monitor their learning process at an abstract level.

Figure 3. TML elements: A) question node; B) start and stop marker C) multimedia symbol; D-E-F) joint line

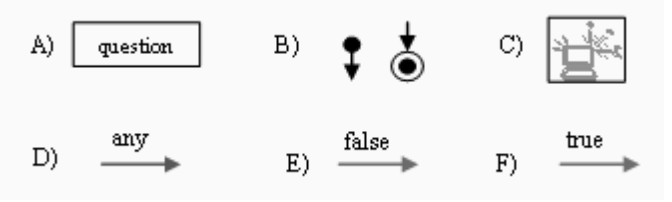
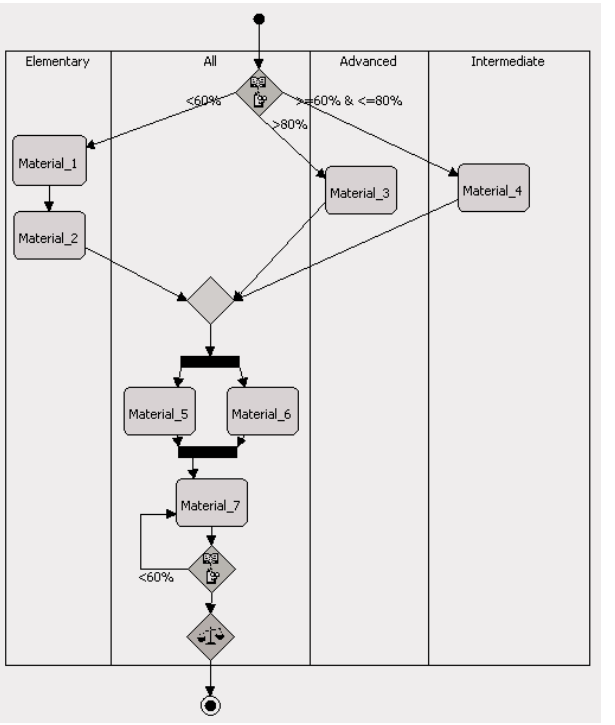


Figure 4. A LAD sentence



The visual sentence shown in Figure 4 represents a set of e-learning activities that have been structured into four swimlanes based on learner knowledge. Student knowledge is assessed using the first test, so if the student has a score less than 60% then the knowledge is considered elementary. Thus, the learner has to study in depth the content presented in Material_1 and Material 2 before going on. When the test result is between 60% and 80%, the contents presented to the learner will be those of Material_4. Finally, Material_3 is presented to the learner with advanced knowledge. After that, there are two parallel learning activities that do not depend on student knowledge. Following Material_7 there is an assessment test, so if one student has a learning deficiency in that self consistent learning activity, then the student must revise it and repeat the associated test. This means that the self-assessment element needs memory to remember the test result.

The instruction designer can describe the learning contents underlying the LAD sentence and refine it by using SCLO. Figure 5 shows a visual sentence describing a navigational schema of a learning content object.

An example of visual sentence from the TML is depicted in Figure 6. Swim lane tags can be used to declare the score that the learner must achieve in order to pass the tests.

SYSTEM PROTOTYPE

In this section we present the SEAMAN system, a prototype based on the

Figure 5. A visual sentence representing a learning content object

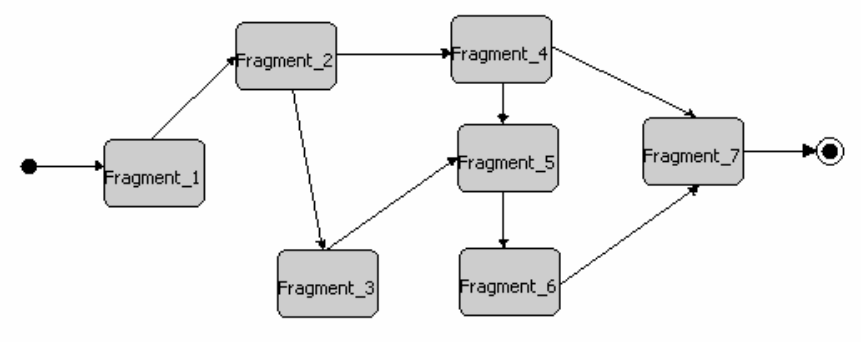
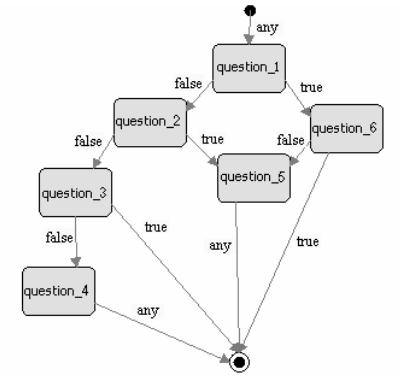


Figure 6. A visual sentence representing a student assessment process



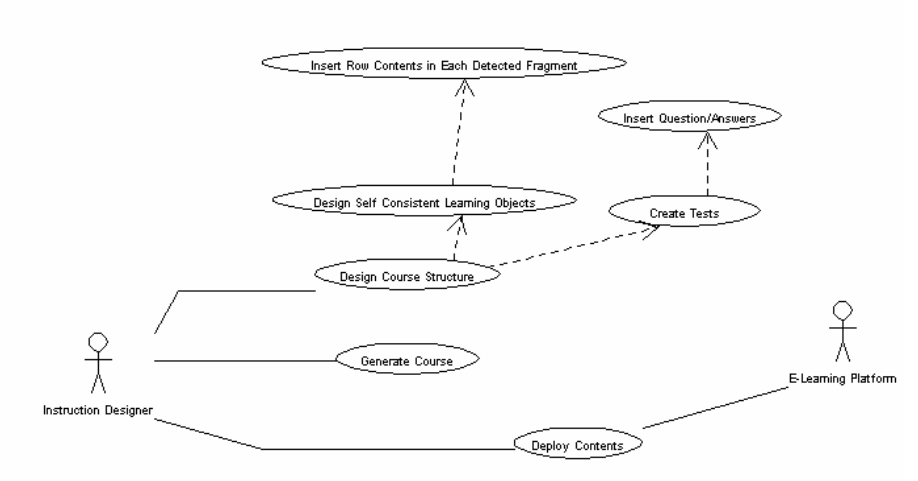
presented approach. SEAMAN assists instructors during the specification and implementation of distance courses and their associated e-learning activities, supporting the delivery of instruction via the Web (Schar & Kriger, 2000; Christoffersen & Christoffersen, 1995). The system integrates modules for several authoring activities, such as knowledge contents, assessment and self-assessment tests. The system can be configured as a centralized application so instruction designers can share and reuse content objects at different granularity levels.

To better understand the functionality of SEAMAN, let us consider the use case diagram in Figure 7 showing the relationships among use cases, instruction designer and e-learning platform.

Two actors can be identified; namely, the instruction designer and the e-learning platform. The former uses SEAMAN to define the entire structures of e-learning courses, contents and tests, using the features provided by the proposed tool. Once the definition of structures and contents is completed, the course based on the specified (asynchronous) activities is generated. The generation process releases instruction contents to be deployed via Web, and will be available using an e-learning platform. It is worth noting that the contents generated are deployable in any SCORM-compliant e-learning platform.

The main modules of SEAMAN as shown in Figure 8 are: the Graphical User Interface (GUI), the Engine, the Information Management module and the Course Delivery module, which have been developed using Java technology. The GUI module is composed of a wizard, three visual environments to implement the proposed visual languages and facilities to reuse and define the e-learning activities. They communicate with the system DB through the Activity Management Module. In order to implement the system Databases we have used Sonic SQL, an open-source pure Java

Figure 7. SEAMAN use case diagram



DBMS, even if the prototype can also be configured to work with Interbase or any other DBMS having a JDBC driver.

Starting from the visual representation and content objects, the Engine module generates the distance course. The aim of this module is to fill templates with contents that we have introduced through the system user interface and to customize Java classes. The Engine module produces two outputs; the first is the distance course in terms of e-learning activities and knowledge contents, whereas the second is a server module configuration file that allows the learning process traceability using a LAD diagram. Using the Delivery module, the instruction designer deploys the distance course and server modules, which will be integrated in a SCORM compliant e-learning platform.

All information required for the generation of e-learning environments is stored in Local/Remote repositories, which contain metadata files to describe each con-

Figure 8. Architecture of SEAMAN system

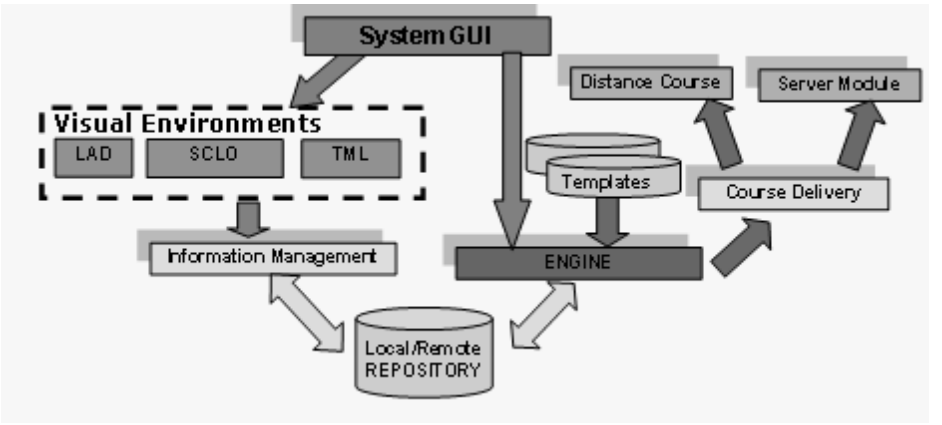
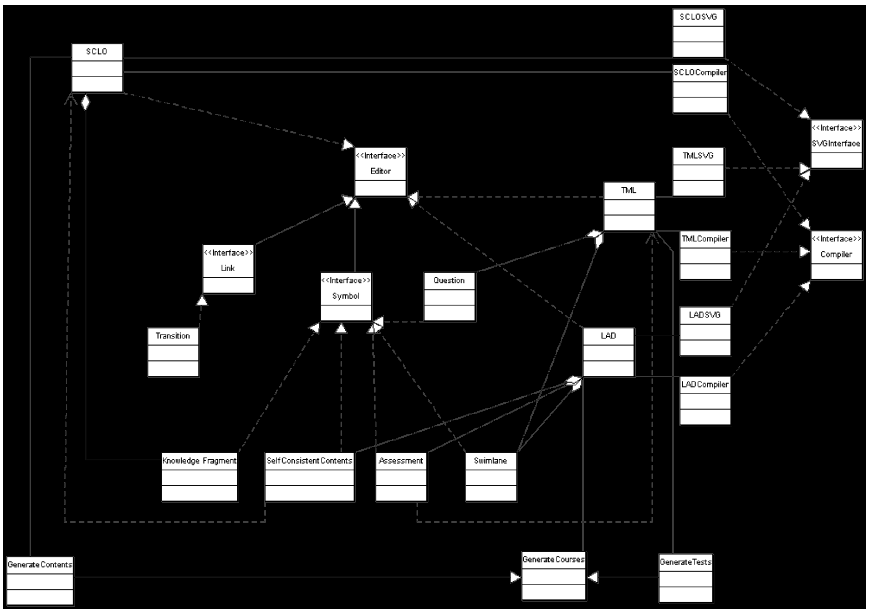


Figure 9. SEAMAN Class Diagram overview



tent object and the multimedia objects composing it. The repositories containing the objects defined with this tool have additional information about the structure. For example, graphical representations of each content object is saved using the Scalable Vector Graphics standard (SVG, 2003), which is XML based.

To support the instruction designer during the design of an e-learning course, SEAMAN integrates three visual programming environments, implementing the visual languages defined above. In order to provide an overview of the system, Figure 9 depicts the UML Class Diagram of such visual programming environments. In the diagram, the relations between the classes implementing the visual environments and those generating the contents of a given e-learning course are also shown. Figure 10 shows the graphical user interfaces of these visual environments.

Output of SEAMAN is an e-learning environment delivered on the Web. Thus, features such as usability, colors and graphical layout become crucial for student welfare and e-learning course success. For this reason, SEAMAN has several predefined graphical layouts that the instruction designer chooses for the Web pages implementing the learning environment. Moreover, the prototype allows us to define new layouts or to customize existing ones.

The e-learning activities generated by the prototype are iteratively navigable through a Web browser. Although the e-learning activities generated using the proposed approach are HTML pages, we need sophisticated technologies that only some browsers support. Thus, SEAMAN works for recent versions of Netscape and Internet Explorer.

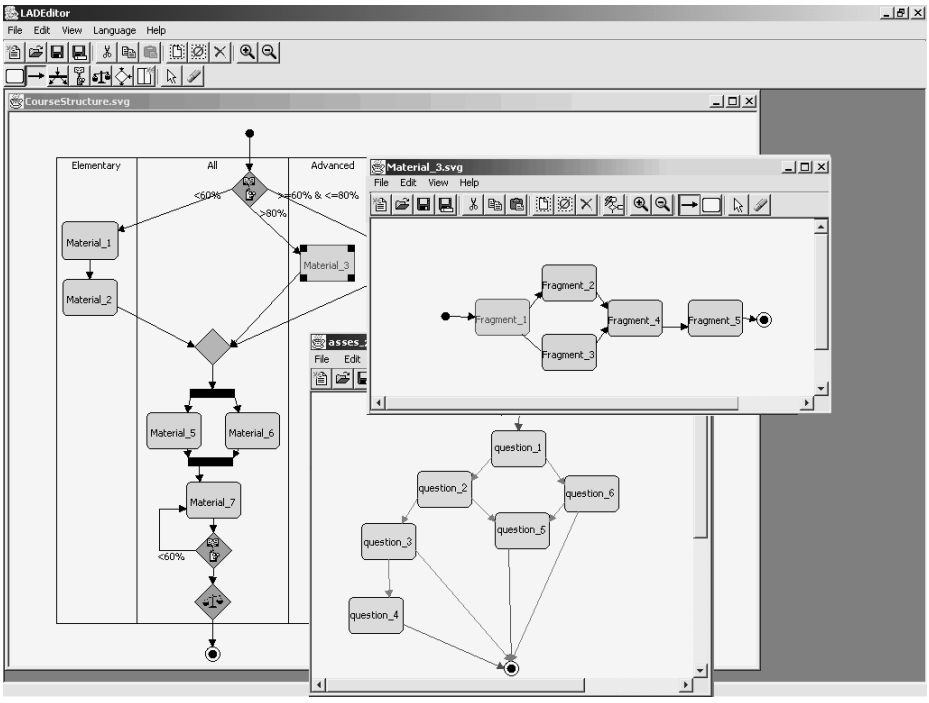
Although the language aims to support the design and development of learning processes, it has also turned out to be a

powerful tool for presenting e-learning activities and for monitoring student progress. Thus, an animation of diagrams allows students to monitor their progresses. Moreover, thanks to the SVG storage format of LAD visual sentences, the student learning process can be visualized using a Web browser with a suitable plug-in. The animation is executed only when the activities of a given distance course are deployed in a SCORM-compliant e-learning platform. Moreover, SEAMAN generates a server module interacting with the LMS. The server module is integrated in the platform when the course is delivered. By coloring the activities green when the student finishes them, we provide a high level representation of the learning traceability.

A SAMPLE APPLICATION

Several lecturers at the University of Salerno have used SEAMAN to create e-learning courses. In particular, in this section we show its use for the design of the *Programming Language Technologies* (PLT) course, belonging to the bachelor's degree in computer science at the University of Salerno. The aim of the PLT lecturer was to design the course to provide students with knowledge and expertise in the design and implementation of compilers. After attending this course, students should be able to design and implement a language compiler. In particular, as a course project, students are required to implement a subset of functionality of the Java language compiler back-end.

Figure 10. SEAMAN's Visual Programming Environments



The main steps to create a course with SEAMAN are:

1. Use the LAD environment to define the course structure in terms of contents and tests;
2. Use the SCLO environment to insert content for each SCLO object mentioned in the LAD sentence;
3. Use the TML environment to define assessment and self-assessment tests in the LAD sentence.

Figure 11a depicts the visual sentence describing the learning process of the PLT course. Figure 11b highlights how the same sentence is presented to the student by using a Web browser with SVG plug-in. The sentence does not provide traceability animation because no e-learning activity has been completed. The two figures show that the course is divided in two parts. The former deepens topics about compilers' preliminary notions and basic tools for developing them (such as Lex and Yacc). The latter part is conceived for designing and implementing compilers. In particular, the main characteristics of the JAVA, C# and C++ compilers are shown. There are no dependencies among these topics, so their content also can be completed simultaneously.

Starting from the objectives and the topics of the course, the lecturer has introduced three tests, two of which are self assessment and one assessment. The first self-assessment test depicted in Figure 11a and Figure 11b regards Lex and Yacc tools. Since these tools are considered vital for the realization of the final course project, the lecturer has introduced a test and a feedback on them. Instead, the run time

Figure 11a. LAD sentence describing.PLT learning process in SEAMAN tool

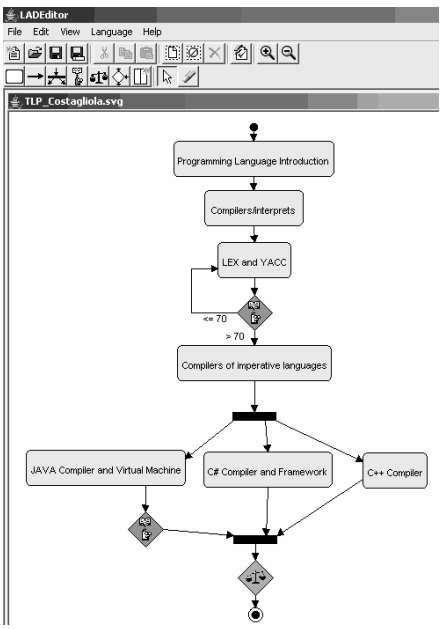


Figure 11b. LAD sentence describing PLT learning process displayed using Internet Explorer

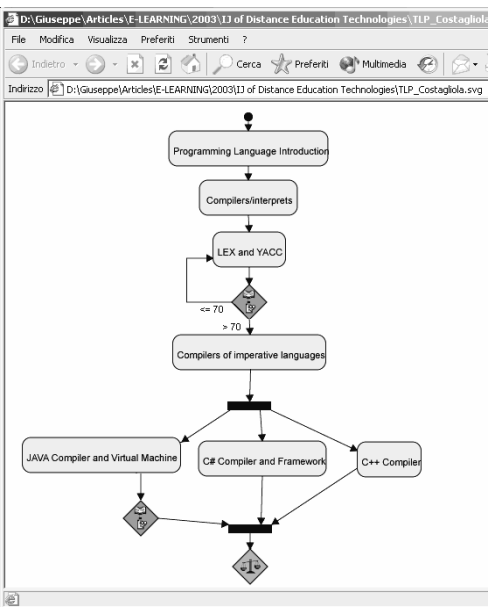


Figure 12. TML sentence describing the test on the runtime environment and the compiler of JAVA

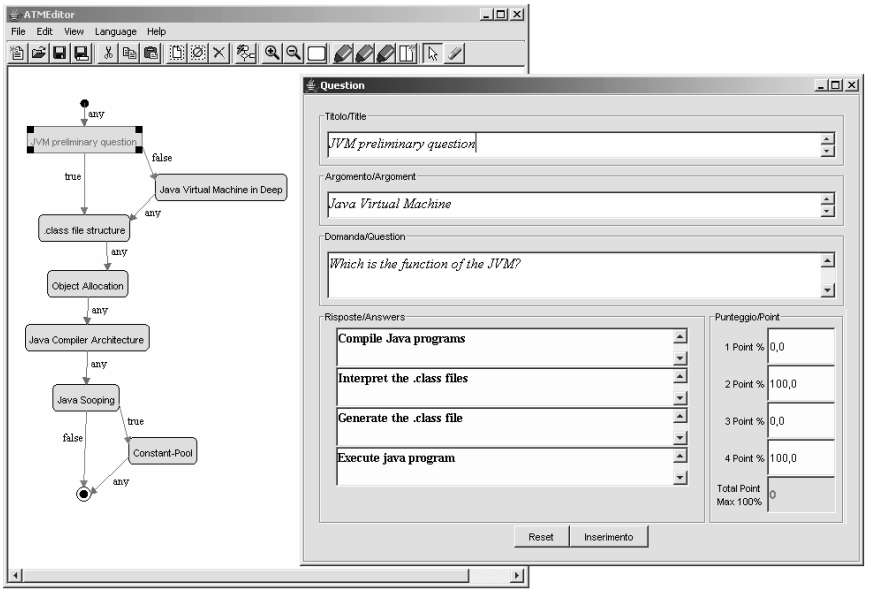
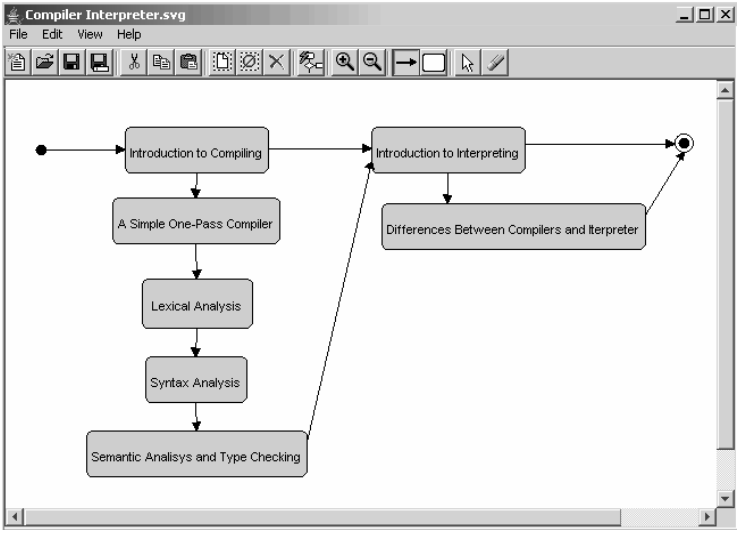


Figure 13. A visual sentence representing the Compiler/Interpreter learning content object



environment and the compiler of the Java language are evaluated by using the second self assessment test depicted in Figures 11a and 11b. The visual sentence describing this test is shown in Figure 12. In this picture we can also see an example of a question with the associated answers. Finally, the lecturer has created a test for assessing the learners' knowledge on the whole course's contents.

The visual sentence in Figure 13 shows the definition of the learning content object "Compilers/Interpreters" depicted in Figures 11a and 11b. The main fragment describes the compilation and interpretation processes, whereas subsequent fragments show the detailed activities.

System Usability

So far, five lecturers at the University of Salerno have used the SEAMAN system to produce a complete electronic version of their course. In particular, they have redesigned the following five courses from bachelors degrees related to computer science: *Programming Language Technologies, Databases, Discrete Mathematics, Fundamental of Physics and Business Administration*. All the lecturers underwent an introductory course of 6 hours on the SEAMAN system and its visual notations. After that, they were asked to use the system on their course, having the possibility to invoke individual tutor support. Once they had completed the design of their course with SEAMAN, they were asked to answer a questionnaire to provide feedback on system usability issues. In particular, other than the course name and the lecturer's familiarity with diagrammatic languages (low, medium, high), the form contained the following three categories of questions: intuitiveness of language symbols and visual sentences, training and usage times, and comprehensive tool evaluation with respect to well-known authoring tools. In what follows we will provide the questions for each category and the associated multiple-choice answers. Then, we will summarize the results of this evaluation in Table 1.

Questions on
Visual Language Intuitiveness

Q1.1)
Do you find the visual symbols of the Visual Languages in SEAMAN easy to understand?

Q1.2)
Do you find the LAD language easy to understand?

Q1.3)
Do you find the SCLO language easy to understand?

Q1.4)
Do you find the TML language easy to understand?

Q1.5)
Do you find the hierarchical organization of the three visual languages easy to understand?

For each of these questions the lecturer could choose one of the following answers:

a. I can understand it very quickly
b. I can understand it with moderate effort
c. I can understand it with a lot of effort
d. It is difficult to understand it

Questions on
Training and Usage Times

Q2.1)
Approximately, how many hours of individual assistance did you need to master the tool?

Lecturers could answer this question by crossing one of the following choices:

a. Less than 10;
b. Between 10 and 20;
c. Between 20 and 30;
d. More than 30;

Q2.2)
How long does it take to formulate a sample visual sentence in LAD language?
Q2.3)
How long does it take to formulate a sample visual sentence in SCLO language?
Q2.4)
How long does it take to formulate a sample visual sentence in TML language?

For each of these questions the lecturer could choose one of the following answers:

a. Less than 5 minutes
b. Between 5 and 10 minutes
c. Between 10 and 20 minutes
d. More than 20 minutes

Comprehensive Tool Evaluation

Q3.1)
Which of the following authoring tools are you familiar with?

Lecturers could answer this question by crossing 0 to 4 of the following choices:

a. Toolbook instructor;
b. Authorware;
c. Dreamweaver;
d. Frontpage

Q3.2)
Do you think visual languages in SEAMAN make e-learning course creation less or more difficult compared to your preferred authoring tool among the ones you selected in question Q3.1?

Lecturers could answer this question by crossing one of the following choices:

a. Much easier
b. Easier
c. Moderately easier
d. More difficult

Q3.3)
Will you use SEAMAN for your other courses?

Lecturers could answer this question by crossing 0 to 4 of the following choices:

a. Yes;
b. Maybe;
c. No;

In general, although the data in the Table 1 refer to a small sample of users, we notice that familiarity with diagrammatic notations seems to facilitate the tool's usage. In particular, non-computer science professors had more difficulties on the LAD language than on the other two visual languages. We suppose this is mainly because although workflows should be familiar in business and many other disciplines, mapping the concept of activity synchronization on real-world problems is not immediate. Moreover, the Discrete Mathematics professor also had some difficulty on the definition of assessment and self-assessment tests with the TML language. More specifically, she found some problems in understanding the meaning of the *true*, *false* and *any* joint links and how they affect the test behavior at run time. In conclusion, the SCLO language appears to be the easiest to understand and use, whereas the LAD requires some effort for people not familiar with activity diagram notations. The hierarchical visual language organization appears to be grasped quite intuitively by all of the lecturers. Regarding the time necessary to enter a visual sentence in SEAMAN, as shown in Table 1, this mostly

Table 1. Summary of SEAMAN's evaluation

Course Name	VL familiarity	Q1.1	Q1.2	Q1.3	Q1.4	Q1.5	Q2.1	Q2.2	Q2.3	Q2.4	Q3.1	Q3.2	Q3.3
Programming Language Technologies	Medium	b	a	a	a	b	a	a	a	a	b,c,d	b	a
Databases	High	a	a	a	a	a	a	b	a	a	a,c,d	b	a
Discrete Mathematics	Low	b	c	b	c	b	c	c	b	c			b
Fundamentals of Physics	Medium	b	b	a	a	b	b	b	b	b	d	c	b
Business Administration	Medium	b	c	a	b	a	b	b	a	b	d	b	a

reflects the familiarity with the given notation, although most of the lecturers took reasonable times. More precisely, for designing the whole course, each lecturer took an average of two work days to compose the LAD sentence with the underlying learning activities, and to add contents, including questions and answers. However, in this count we did not take into consideration the time spent for turning course contents in the electronic format, which required a considerable amount of time for the *Discrete Mathematics* course, since most of the material was only available in hard copy.

Finally, looking at the feedback of questions in the third group, it seems that the use of visual languages in SEAMAN makes course creation somewhat easier as opposed to traditional authoring tools, and that all of the lecturers involved in this experiment plan to possibly use SEAMAN in the future.

Achievements and Future Work

Academic and commercial e-learning authoring tools (Apple, Nygren, Williams, & Litynski, 2002; Campbell & Mahling, 1998; Douglas, 2001; Goodyear, 1997; Muraida & Spector, 1997) use the basic concepts of the Multimedia Software Engineering (MSE) (Christoffersen & Christoffersen, 1995). This is a new frontier for both Software Engineering (SE) and Visual Languages (VL) (Bell & Jackson, 1992; Campbell & Mahling, 1998, Costagliola et al., 2003). As it has happened for these fields, the e-learning field also can benefit from the use of visual languages to simplify the work of an instruction designer. To this sake, in this paper we have presented three visual languages for defining several stages of the distance course design process. The languages have been implemented within SEAMAN, a prototype conceived to allow instruction designers to more easily generate learning environments and enhance their welfare. Experimental results have shown that the use of visual languages can encourage the design of distance courses, as opposed to what happened before with tools using more rudimental interaction paradigms.

In the future we aim to extend the visual languages proposed in this paper. This has a two-fold goal. On one hand, we aim to introduce less technical and more metaphor-oriented visual languages to provide further abstraction in the e-learning course design process, and consequently increase system usage among non-technical teachers. On the other hand, we aim to extend our visual languages to enable the modelling of additional aspects of the e-learning process. Special interest deserves the specification and the realization of collaborative activities, in order to encourage cooperative and problem-based learning (Laister & Koubek, 2001). In this way, groups of students can work together to solve problems while keeping their diversities. As a result, the learning environments will encourage individual accountability, prompt feedback and high self-expectations.

REFERENCES

Advanced Distributed Learning (ADL). (2003). *SCORM present*. Retrieved online: *www.adlnet.org*

Apple, D.K., Nygren, K.P., Williams, M.W., & Litynski, D.M. (2002). Distinguishing and elevating levels of learning in engineering and technology instruction. *Proceedings of the 32nd ASEE/IEEE Frontiers in Education Conference* (pp. 6-9).

Bell, M.A., & Jackson, D. (1992). Visual author languages for computer-aided learning. *Proceedings of the IEEE Workshop on Visual Languages* (pp. 258-260).

Campbell, J.D., & Mahling, D.E. (1998). A visual language system for developing and presenting Internet-based education. *Proceedings of the IEEE Symposium on Visual Languages* (pp. 66-67).

Christoffersen, R.D., & Christoffersen, A.H. (1995). Instructional system design: Its role in coast guard training and how it can influence the development of training materials. *Proceedings of the Professional Communication Conference, IEEE International* (pp. 39-43).

Costagliola, G., Ferrucci, F., Polese, G., & Scanniello, G. (2003). A visual language for designing and presenting e-learning activities. *Proceedings of the IEEE First International Conference on Information Technology: Research and Education* (pp. 630-634).

Cynthia, J., Finelli, M., & Wicks, A. (2000, May). An instrument for assessing the effectiveness of the circuits curriculum in an electrical engineering program. *IEEE Transaction on Education*, 43.

Designer's Edge. (2003). *The industry standard instructional design tool*. Retrieved online: *www.allencomm.com/products/authoring_design/designer/*

Douglas, I. (2001). Instructional design based on reusable learning object: Appling lessons of object-oriented software engineering to learning system design. *Proceedings of the ASEE/IEEE Frontiers in Education Conference*.

Ferrucci, F., Tortora, G., & Vitiello, G. (2002). *Visual programming*. Encyclopaedia of Software Engineering. John Wiley & Sons.

Goodyear, P. (1999). Seeing learning as work: Implications for understanding and improving analysis and design. *Journal of Courseware Engineering, 2*, 3-11.

Jones, D., Shirley, G., & Lynch, T. (2003). An information systems design theory for Web-based education. *Proceedings of the IASTED International Symposium on Web-based Education*.

Kasowitz, A. (1997). Tool for automating instructional design. *ERIC e-Learning House in Information Technology in Education*. Retrieved online: *http://ericit.org/digests/EDO-IR-1998-01.shtml*

Laister, J., & Koubek, A. (2001). Third generation learning platforms requirements and motivation for collaborative learning. *European Journal of Open and Distance Learning*.

Muraida, D.J., & Spector, J.M. (1997). Automatic design instruction. S. Dijkstra, N. Seel, F. Schott, & D. Tennyson (Eds.), *Instructional design: International perspectives* (vol. 2). NJ: Lawrence Erilbaum.

OMG Group. (1993). OMG unified modeling language specification. Retrieved online: *www.rational.com/media/uml/post.pdf*

Safoutin, M.J., Atman, C.J., Adams, R., Rutar, T., Kramlich, J.C., & Fridley, J.L. (2000). A design attribute framework for

course planning and learning assessment. *IEEE Transaction Educational, 43*, 188-199.

Scalable Vector Graphics (SVG). (2003). XML graphics for the Web. Retrieved online: *www.w3.org/Graphics/SVG/*

Schar, S.G., & Kruger, H. (2000). Using new learning technologies with multimedia. *IEEE Multimedia*, 40-51.

Vrasidas, C. (2002). A systematic approach for designing hypermedia environments for teaching and learning. *International Journal of Instructional Media*. Retrieved online: *http://www.cait.org/vrasidas/hypermedia.pdf*

Please provide bios