Int. J. Engng Ed. Vol. 26, No. 1, pp. 111–125, 2010
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2010 TEMPUS Publications.

Computer-Based Knowledge, Self-Assessment and Training*

MARKO ČUPIĆ, ŽELJKA MIHAJLOVIĆ
University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Electronics, Microelectronics, Computer and Intelligent Systems, Unska 3, 10 000 Zagreb, Croatia.
E-mail: zeljka.mihajlovic@fer.hr

We present the design, technical issues and use of a software environment for learning and educational assessment. Our proposed system offers individualized tasks for each student and allows variations and modifications of the tasks (with instructor supervision). The system is designed to support distance learning, but it can also support and enrich education in a traditional classroom. Using application-based computer graphics tasks, we present various potential uses of the proposed software environment. Computer graphics tasks are highly demanding for interactivity, visual presentation and simulation. The proposed design addresses the most demanding graphics challenges. We demonstrate that the proposed framework applies to different courses and that this mode of learning is highly motivating for students.

Keywords: educational technology; programming environments; unsupervised learning; computer graphics software

1. INTRODUCTION

DURING THE LAST TWO DECADES, much effort has been made to incorporate computers, electronic devices, and Internet technologies into educational settings in an attempt to provide a new, rich and integrated platform with a variety of systems and content that helps users acquire new knowledge. At first, existing Internet technologies were applied in education in an attempt to enrich traditional courses given by human instructors. For example, chats enabled synchronous user-to-user and user-to-instructor communication, forums and e-mails enabled asynchronous communication, web applications offered a means to distribute course materials (such as Microsoft PowerPoint presentations, PDF documents, or audio and video materials). Popular products that provide such services include Moodle [1] and BlackBoard [2]. In addition, a product focused on course management support and laboratory exercises is described in [3].

Today, however, we expect available technologies to offer much more. Available technologies have a wide range of modes that can enrich education with visualization elements, virtual environments [4], remote real laboratories [5], simulation [6], interaction and collaboration environments in the context of a real or virtual university. Such technologies make learning attractive, exciting and effective. In particular, they involve students in an active and interactive mode of learning, not only in course materials, but also in the design process for those materials [7]. Learning through game playing is also emerging as a successful approach to education [8].

Instead of speaking of PDF or PowerPoint files, which are understandable only by humans, now we can speak about learning objects: objects enriched by metadata that are suitable for learning, education or training, as specified by the Learning Object Metadata specification [9], Sharable Content Object Reference Model specification [10] or by IMS specifications [11]. Objects described using these specifications are suitable for computer processing. Standards such as IMS Simple Sequencing and IMS Learning Design allow us to specify the teaching process. Specifications such as IEEE Public and Private Information for the learner (PAPI) [12] and IMS Learner Information Package [11] enable us to describe the users' model (current users' knowledge and abilities). When properly used and combined, these specifications permit a software system to select what, how and when should be offered to the user during learning. An example of such a system is ALE [13].

An important part of this education paradigm is the ability to provide a personalized learning experience [14], [15]. For example, Dolog et al. in [15] illustrate this with a scenario in which a person was given a task to create a module for an e-banking application for a local bank. This task required knowledge of the Java programming language and an understanding of banking operations and security considerations, as well as additional specialized knowledge. Given the current user's profile and other constraints, such as the time available for learning and the budget, the software was tasked with recommending a per-

* Accepted 22 July 2009.

sonalized selection of learning materials and learning activities that would contribute most to the user's learning goals and help the user do the given task.

Today, a large quantity of learning material must be distributed among many systems, and that materials as well as users can migrate from one institution that uses a particular set of systems to another that uses different systems. In a distributed environment, such systems must be able to share, query, transfer and consider user profiles and learning objects. Aroyo et al. [16] described the current state of interoperability for adaptive learning components and concluded that, with respect to interoperability either between systems or between formal models, current solutions are unsatisfactory. Kravcik [17], [18] discussed usage of the semantic web to overcome some of the existing issues.

New approaches to e-learning platforms are based on services instead of on a monolithic structure [19]. Such platforms will support federated exchange between services, various levels of interoperability and service composition through orchestration and choreography [20]. Service composition and choreography will allow for dynamic discovery and assembly of e-learning services in order to satisfy user requirements [19].

An often-stated goal that should enable broader usage of educational systems is to improve usability while minimizing the complexity of the authoring tools. A successful example is the WINDS project [21], [13] with which authors without programming skills can produce adaptive course materials by specifying declarative knowledge for adaptation by means of pedagogical metadata.

Knowledge assessment is an important part of every educational system. Educational systems can make use of such assessments to check whether users have learned particular concepts, and update users' profiles accordingly. When used as a supplement to traditional courses taught by instructors, computer-based knowledge assessment could be used to save work that would otherwise fall to human graders, speed-up the grading process and significantly improve objectivity. Many popular software tools such as MOODLE [1] offer knowledge assessment through quizzes with questions written by the instructor. The software presents either all or a random selection of the prepared questions to users during assessment. Considering the fact that not all questions are equally difficult, algorithms for question creation and selection are still being developed (e.g. [22], [23]). Tovarozek et al. [24] describe a method for generating adaptive assessments that is suitable for well-structured domains. Assessment as a part of an e-learning platform is described by the IMS Question and Test Interface Specification [11].

Brusilovsky [25], [26] and Pathak [27] suggest the use of individualized questions for self- assessment of programming knowledge. They show that parameterized code fragments may be used for student assessment, and can lead to significant improvement in students' knowledge. The important observation to come from their investigations is that students themselves value the system highly as a very helpful learning tool.

Our focus is on producing a scalable solution for generating individualized questions that include individualized multimedia content, complex generation of algorithms and work in cooperation with other systems to actually prepare the questions. The first version of the described framework and prototype implementation was introduced in [28]. We present an improved and extended version, which is also open-source. Also, we describe the application and usage of the system we developed based on our framework in a field that inherently requires a rich multimedia environment for presenting and responding to questions: computer graphics. The integrated system requires strong support from our software framework. While the system individualizes parameterized graphic questions and exercises that require interactive graphic solutions, the instructor can adjust the parameters range and consequently tune the complexity of the assignment. The parameterized approach is also an important concept for personalized access to learning environments [15]. Namely, the personalized approach anticipates complexity control, and our framework provides it directly. Our framework should also enable easy inclusion of various algorithms, such as in [23]. Unfortunately, this approach implies incompatibilities with specifications such as IMS QTI Question and Test Interoperability [11]; the reasons for this are explained near the end.

2. REQUIREMENTS AND KEY IDEAS

Let us first discuss some of the important requirements for the software environment. Although we focus on student assignments and exams, note that we are not just interested in simple problems. The idea is to create dynamic and versatile problems that emphasize higher order thinking skills rather than just drills and practice. Thus, whenever possible, we include some parameters to change not only the numbers in a problem, but also to dynamically choose between similar problems. During the learning process, students may practice on assignments as many times as they want. Instructors can observe their progress through records of this process. Another important requirement is that of auto-evaluation. For some problems, this is very difficult to arrange: some tasks have pictures for results, and for others the procedure is equally important if not more than the final result. Adaptability is another important goal. The system must guide students to successive tasks based on the correctness of their previous answers, influencing the learning process adaptively.

The key components of the system should be designed independently of the presentation technology: therefore, we use a multi-technology platform. It is important to bear in mind during the design process current and future tasks and requirements, as well as to design basic components that are independent of the presentation technology.

2.1 Dynamics capability

Dynamics capability enables identification of problems that can use a template for generating questions from problems with statically preloaded question text based on the problem-type and the preloaded possible answers. Typical ABC questions supported in many popular e-learning systems do not have this capability; a user must preload the question text and possible answers, the order of which can then be randomized. Randomization capability means that the same question usually will be presented with a random order of answers to different students.

The simplest form of a problem with dynamics capability is a problem that can be stated in a pseudo-language as follows: 'What is the result of addition of $\{\$a\}$ and $\{\$b\}$?' A correct solution would be specified as '$\{\$a\}+\{\$b\}$', and constraints may be written as 'a, b are integers from interval [0, 50]'. More advanced dynamics capability is associated with problems that can dynamically generate multimedia objects, such as individualized images and incorporate them into questions. Another example of the use of dynamics capability is the creation of a scene with objects specified by a predefined set of parameters, such as the initial and transformed positions of the object. In that case, the number and range of parameters for transformations should be defined to create a figure that describes the problem. For each student, a different scene is created by randomization of the parameters. Dynamic capabilities are used not only in the presentation of problems, but in their solutions as well. The creation of a BSP (Binary Space Partitioning) tree for a given scene is an example of an interactive and dynamic construction of solutions for a particular problem.

2.2 Auto-evaluation

Auto-evaluation capability means that the software can automatically evaluate an answer and determine its correctness. For the most complex problems, sometimes a simulator is required to evaluate the result. An important advantage of auto-evaluation is that it gives students feedback on the correctness of the results that they have achieved. Another advantage of auto-evaluation is objectivity in the process of validating assignments; in addition it relieves instructors or teaching assistants from the job of grading the responses. A representative problem that cannot undergo auto-evaluation is an essay-like problem, where the student writes the answer in natural language. Some problems where the student must enter textual answers can have auto-evaluation capability, but only in a limited sense. If the answer is limited to only a few predetermined words, the software can verify the answers by using regular expressions.

2.3 Adaptability

When presenting the problems, there are also some important concerns about their ordering. In short, what problems will the assignments constitute, how many questions will be asked and will the correctness of a previous question have any influence on the selection of the next question? All of these concerns relate to the same issue: adaptability. If the assignment is adaptable, on what grounds is this adaptability based? Adaptability can be based on a simple algorithm or on more complex artificial intelligence algorithms and methods.

When planning our framework, we chose not to fix any of these issues and capabilities, but to rather build a framework capable of supporting all of them.

2.4 Multi-technology presentation

A particular challenge is to allow mobility and support a diversity of presentation technologies. These capabilities are frequently lacking in other software products. We would like to provide a user with a variety of presentation options. Presentation technologies could be provided through a local GUI-based application, through a Web browser or through a cell phone. This issue is important for two reasons: first, we would like

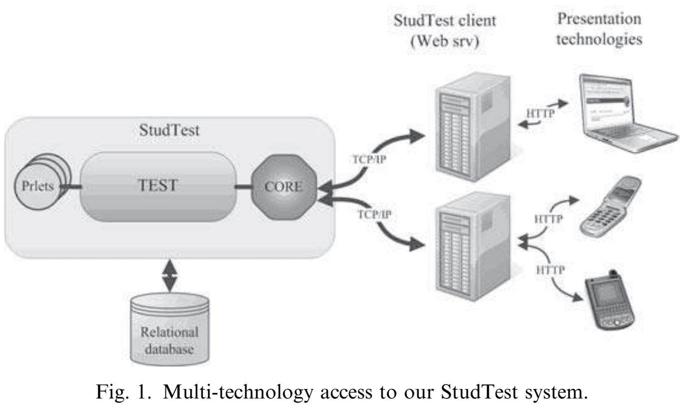


Fig. 1. Multi-technology access to our StudTest system.

our architecture definition to be technology independent; second, we recognize that the choice of technology may determine possible user responses. Depending on the chosen presentation technology, an answer could be submitted by clicking or selecting something, by entering text or a number, or perhaps by drawing. Figure 1 demonstrates the multi-technology access to our system, *StudTest*. *StudTest* supports different presentation technologies over widely adopted protocols, such as HTTP and TCP/IP.

3. PRACTICAL CONSIDERATIONS

Course policy is another issue that must be taken into account. A typical example is a policy such as 'the student must pass test *X*, where the number of attempts is unlimited', or 'students are not allowed to take test *Y* if they have not passed test *X*'. Also, some courses have the following policy: 'Students can solve test *X* as many times as they want; we will grade them by their last attempt'. Such a policy is commonly used to ensure that students have learned the course material. Another example of a course policy is 'Test *X* can be taken for no longer than 15 minutes'. Different course policies may be applied during the learning and exam phases.

Security is another important issue: that is, who will be able to access the tests, and when? If testing is supervised for the entire cohort of students, although students are tested in small groups (e.g. there are a limited number of available computers), care must be taken to disallow access to the tests for students who are not under staff supervision. Today, this is typically accomplished by password protection. To disable password leakage to outside students (e.g. through cell-phones), IP-based control can also be used and/or passwords can be changed on a regular basis.

The last issue that we will analyse is scalability and handling of heavy loads. The system should be designed to handle a large number of users (the order of thousands). But depending on the specific course organization, situations are possible where many of the students using the system simultaneously during a time-constrained testing can generate heavy peak loads. In those conditions, it is critical for the system to have short response times. This can be achieved in two ways: by building a clustered system with load-balancing support or by building a system based on asynchronous operation that can postpone less important operations during heavy loading conditions (or both). In order to offer the possibility of both solutions, we defined an asynchronous operations-based framework that can easily be clustered.

4. MODEL DEFINITION

We first define the core elements of our architecture and describe the basic concepts and inter-

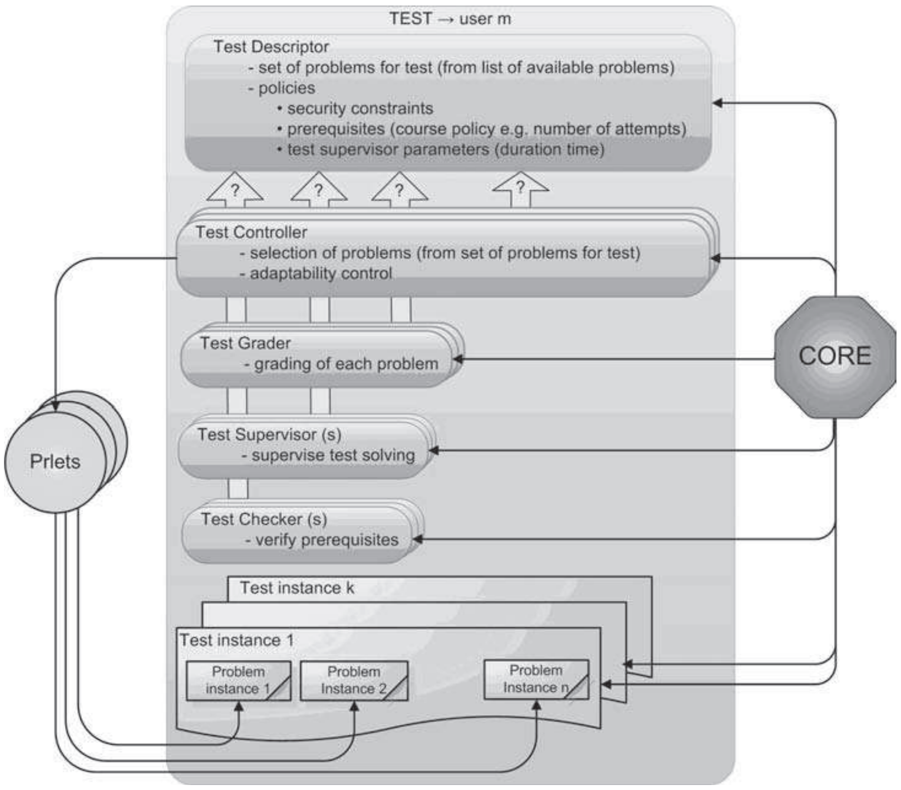


Fig. 2. Core elements of the StudTest architecture.

actions between them. We define Test as a collection of all students' attempts to solve given assignments.

4.1 Core elements

The core of the system connects the TestController, TestGrader, TestSupervisor and TestChecker with the TestDescriptor (Fig. 2). The TestDescriptor describes a test. Through the Core, the instructor defines the main objectives for the test. From the list of all available problems, the instructor selects a set of problems appropriate for the particular exam or by stating examination objectives. The number of problems in that set is greater than or equal to the number of problems in TestInstance that will be generated. If the number is larger, problems can be randomly chosen. TestController determines which problems are to be included according to the specification. In addition, TestController, according to policies defined in TestDescriptor, determines whether the test is adaptive. TestGrader determines the score according to the specified evaluation (details will be further explained). TestDescriptor will enable inclusion and configuration of available security constraints.

Practical considerations are included through TestSupervisor and TestChecker. TestSupervisor supervises test solving. Rules are defined in the policies of the TestDescriptor. An example of a rule is a time limit. TestChecker verifies prerequisites. For example, the number of attempts to create TestInstances could be a specified prerequisite.

After the test description is ready, the student may go through TestController to initiate a TestInstance. The TestInstance is a concrete test that will be presented to the student. For each student and TestDescriptor, the TestInstances are grouped into collections by means of Test. Thus, the Test is a wrapper that is constructed for each student. Test can be visualized as a folder containing all of the student's attempts to solve the specified TestDescriptor.

The most crucial part of the test design concerns ProblemInstance generation. Our framework heavily relies on support of the concept of prlets (pronounced 'pearl-ets'), which we introduce here (the term is inspired by servlets, which are currently widely accepted and used as a core Java-based technology for Web applications, standardized by Sun (JSR-000053)). It is also worth mentioning that a similar attempt was made at Ramapo College of New Jersey. That team introduced problems with more than the usual capabilities, known as problettes, that can be used in most Java enabled Web browsers in form of Java Applets [29].

4.2 Prlets

To offer a variety of possibilities and capabilities, we have defined the concept of Prlet, and constructed the rest of the framework to be a Prlet container: a component-based environment that executes Prlets and supports pluggable objects. The general idea behind Prlets is that they represent pluggable components that have public names and can be referred to globally, which means they can also be easily shared. They contain the complete logic needed for problem editing, instantiation and potentially evaluation. This framework operates with several categories of objects, as follows. The *Prlet* is a component composed of the following objects: ProblemGenerator, one or more ProblemEditors, ProblemInstantiator and ProblemEvaluator, as shown in Fig. 3.

The ProblemGenerator stores basic information about a Prlet, including its public name, problem type and whether or not it can evaluate answers automatically. The ProblemEditor allows for customization of a problem template based on concrete questions (to be introduced later as ProblemInstances). Through ProblemEditor, a instructor specifies ranges for parameters for particular problems, making them more or less complex. A parameter could define, for example, the number of elements in a scene. Obviously, a scene with more elements presents a more challenging task than one with fewer. ProblemEditor also includes supported technology identifiers. Technology identifiers influence problem presentation. Editing of problem template parameters can be supported in a narrower range of technologies where standard HTML is mandatory; however, a new editor must be written for each technology.

The ProblemInstantiator generates concrete problem based on the current parameters of the problem template. Most of the power of the presented framework lies exactly here—problem instantiation allows the integration of separate components (e.g. communication with other servers on the Internet, Web services for help, etc.) that can be used to create a new instance of a problem. Since ProblemInstantiator knows the type of problem it creates, all the necessary data imposed by that type will be stored in ProblemInstance's repository (private data storage).

The ProblemEvaluator evaluates and generates comments on user solutions, determines their

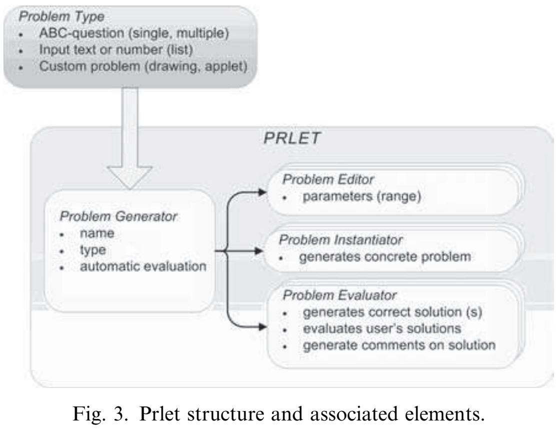

Problem Type
• ABC-question (single, multiple)
• Input text or number (list)
• Custom problem (drawing, applet)
PRLET
Problem Generator
• name
• type
• automatic evaluation
Problem Editor
• parameters (range)
Problem Instantiator
• generates concrete problem
Problem Evaluator
• generates correct solution (s)
• evaluates user's solutions
• generate comments on solution

Fig. 3. Prlet structure and associated elements.

correctness using a predefined measure and generates the correct solution in the case where solution generation is computable, supported and no correct answer was provided by the user. Due to this division of labour, evaluators not only implement complex algorithms themselves, but may also use other resources for evaluation purposes, such as contacting other servers on the LAN or the Internet in order to use prepared clusters for necessary calculations, and so on.

ProblemType represents a technique for presenting a problem to a user. Typical problem types are single–correct–ABC-question, multiple–correct–ABC–question, input–textOrNumber–question, input–listOf–textOrNumber–question and CustomProblemPanel. The latter was defined in order to support problems that will be presented uniquely in a graphical user interface. The system offers rich tools such as drawing for the solution input. In our framework, we separate this information from the Prlet itself in order to distinguish type-presentation issues from problem-logic issues, leaving only the logic issues as a part of Prlet. This separation also enables implementation of multi-platform presentation capability. Namely, when the user's client contacts StudTest (Fig. 1), as a part of the handshaking process it must send a TechnologyIdentifier telling the StudTest what test presentation technology the client supports. Based on this parameter, StudTest can than select for each ProblemType (Fig. 3) an appropriate ProblemRenderer component, which knows how to appropriately present that problem type to the user using the user's own technology.

Hence, the ProblemRenderer is a component used to present problems of a specified type using the selected technology. Thanks to this separation of parameters, all that is needed is to implement added support for new technologies is an additional set of renderers for the specific technology to be added; that is, no Prlet needs to be aware of the change.

4.3 Protocol

To allow for customization, StudTest defines an assessment protocol based on finite automata, shown in Fig. 4, with a typical scenario given in Fig. 5.

When a student first requests a test by following a link in web browser, StudTest creates a new test instance for that student. At that point, the test instance is not initialized. Next, StudTest starts the preparation phase. The TestController selects an initial set of problems and instantiates them (Fig. 4 and Fig. 5). When all problem instances have been instantiated, a test instance is prepared. Next, checkers defined in the test descriptor are consulted to see if the student can access the test instance. Access can be denied if, for example, the student's request comes from a disallowed IP address, or if the time for the test has not yet come. If any of the checkers deny access, the test instance state will remain uninitialized. If all the checkers grant access, the test instance is enabled. All of the defined supervisors are then advised that testing has begun. Once the first question is selected and displayed, the test instance transitions to the state 'Solving in progress'. StudTest waits for the student's solution.

When the student enters a solution and requests the next problem, all supervisors are consulted to see if the interaction is still allowed. The student solution is saved, and the next problem is displayed. TestController determines which actions will be available to the student. A typical set of actions will include 'Next problem' and 'Previous problem' (if such exists), direct navigation to all test problems, test suspension and 'Finish test'. If test suspension is requested, the test instance will be frozen rather than proceeding to the next problem. This would be allowed in cases where a student is permitted to solve the test over a few days.

When the student solves the test and triggers the 'Finish test' action, the test instance transitions to the 'Solving finished' state. Evaluators for all problem instances are started, followed by configured graders. When all problem instances have been evaluated and graded, the test controller calculates the total score for the test instance, and determines whether the student passed or failed.

Since StudTest offers such a fine-grained examination protocol, other examination scenarios are possible. For example, given an appropriate test controller, a limited number of problem instances can be selected during the preparation phase. Upon completion of the preliminary assessment, TestController can immediately begin evaluating them and select additional problems based on the

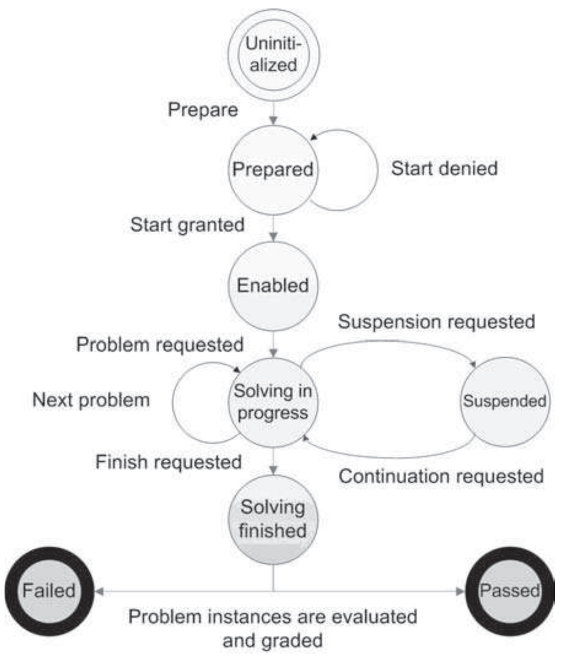


Fig. 4. Simplified examination protocol.

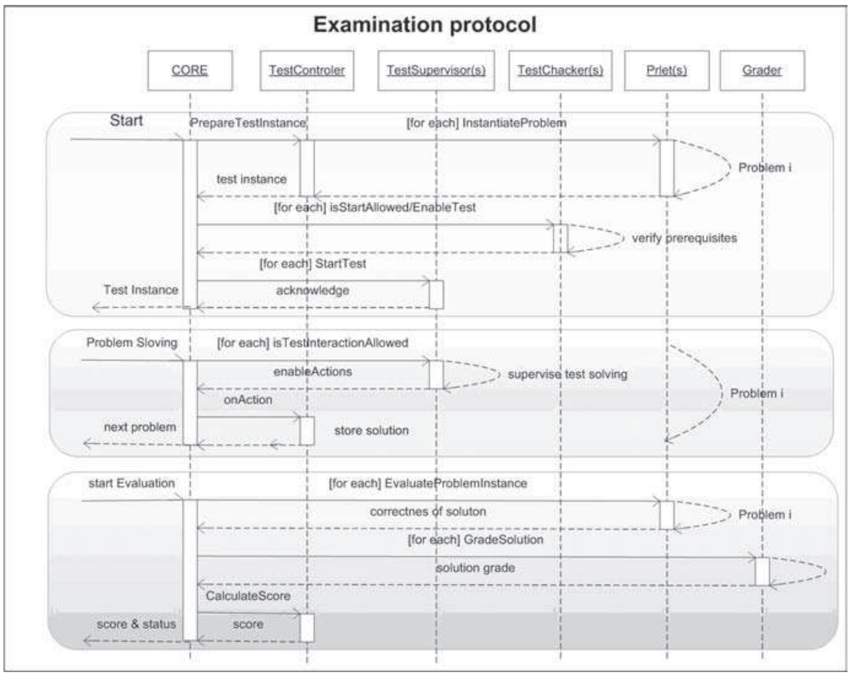


Fig. 5. Typical examination scenario.

evaluation result, thus creating an adaptive test. Unfortunately, this protocol is incompatible with IMS QTI Question and Test Interoperability. The reason for this is that the general idea behind StudTest is to allow flexibility such that skilled programmers can easily add new test checkers, test supervisors, test graders and test controllers. New problem types can easily be defined and added, enabling and supporting development of new problems suited for students and assessment of their knowledge. Programmers of average skill can easily develop new problems by using already-defined problem types. However, in order to do this, some sort of programming knowledge is necessary, making XML-based problem descriptions, which comprise the foundation of IMS QTI, inadequate.

5. IMPLEMENTATION OF PROBLEMS BASED ON StudTest FRAMEWORK

5.1 Implementation of computer graphics problems

The StudTest framework was implemented in the Java programming language. We chose to use Java technology for two reasons: first, to our knowledge, only Java offers a stable and portable platform for application development. Because of its broad acceptance and the existence of Virtual Machine implementations for almost all widely-used operating systems, Java is the optimal choice. The second reason is that our goal is to support graphically-rich problems with complex user interfaces and complex methods for solution input. Considering the tendency to move assessment systems to the Web and HTML, it is again clear that the only portable platform for these purposes is offered by Java Applets. An additional reason to use Java is that some kind of scripting language is required in order to support dynamic problems; today, JavaScript is often used for that purpose. JavaScript was initially introduced as a simple scripting language for simple client-side computations and event handling, and was never meant to be a full-fledged object-oriented programming language. We have decided to base all of our scripting needs on a regular and widely accepted object-oriented programming language [30] with modern language constructs and built-in support for concurrency.

5.2 Some computer graphics problems

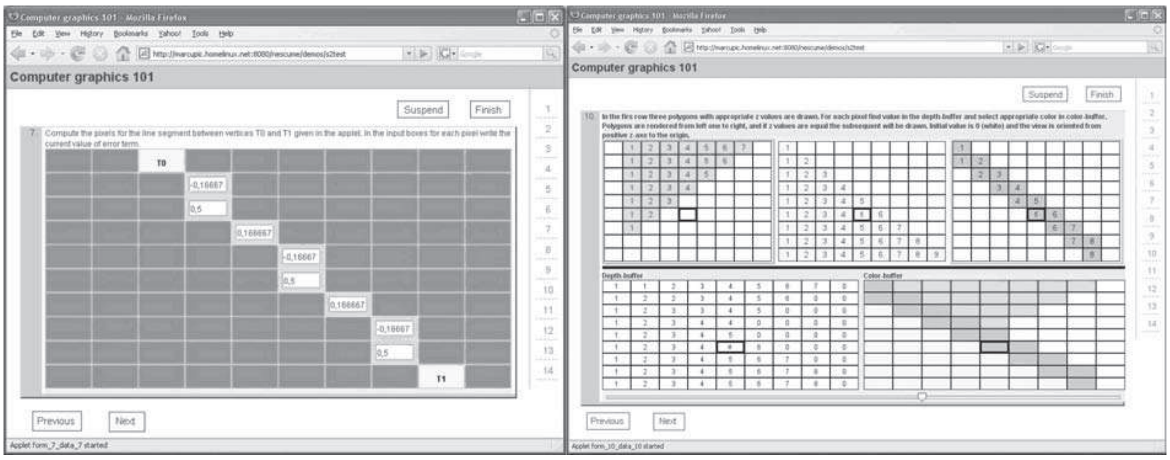
Many elements of standard computer graphics problems were implemented based on the StudTest framework. First, we present the use of the StudTest framework to test for some fundamental computer graphics algorithms such as Bresenham's line drawing algorithm and depth-buffer technique [31].

Bresenham's line algorithm determines which points in a two-dimensional discrete raster space should be drawn in order to form a close approximation of a straight line between two given points T_1 and T_2. Each pixel is represented as enlarged. After selection of a pixel, a value for this pixel can be assigned with an input box. Pixels are chosen according to the integer value corresponding to the pixel centre which is the closest to the ideal line between T_1 and T_2. For each raster point that is selected as a line point, an input box is opened and an error value determined by the algorithm can be entered. Hence, to determine the error values, a student should follow the Bresenham's algorithm by hand and obtain the required values. Evaluation of correctness is performed based on the selected pixels and their actual error values. In Fig. 6. on the left side, the Bresenham's line drawing algorithm is presented in the real environment. This particular exam had fourteen problems and the Bresenham's line drawing algorithm is presented as seventh. A student may choose the next problem directly by selecting the problem number on the right or by selecting Previous or Next. Current solutions may be saved and continued later by choosing Suspend. Finish will begin the evaluation of the solutions.

Another simple example is the depth-buffer technique. In the first row, three polygons with appropriate depth (z) values and different colours are drawn. The goal is to find a value in the depth-buffer for each pixel and to select the appropriate colour in the colour-buffer (second row). The initial value is zero (white) and the view is oriented from the positive z-axis to the origin (Fig. 6. right). Selection of a pixel in the depth buffer changes its depth value and selection of a pixel in the colour buffer changes the colour between four possible options. Each pixel may be selected using the mouse or a slider at the bottom of the screen. Although these are simple tasks, they are nevertheless important to grasp for understanding basic computer graphics techniques, and are appropriate for demonstration of the problem implementation in the StudTest framework.

5.3 BSP Tree

The BSP (Binary Space Partitioning) tree partitioning technique is one of the most successful space partitioning techniques, since it allows both object modelling and classification via a single structure. To learn how to create and handle BSP partitioning trees, three problem types were posed. The first was to create a tree structure for a given scene, as given on the left side of Fig. 7. When the


Computer graphics 101 - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://marcupic.homelinux.net:8080/rescune/demos/s2test
Computer graphics 101
Suspend
Finish
Previous
Next
T0
T1
1 2 3 4 5 6 7 8 9 10 11 12 13 14

7 Compute the pixels for the line segment between vertices T0 and T1 given in the applet. In the input boxes for each pixel write the current value of error term.

-0,16667
0,5
0,166667
-0,16667
0,5
0,166667
-0,16667
0,5

Applet form_7_data_7 started

10 In the firs row three polygons with appropriate z values are drawn. For each pixel find value in the depth-buffer and select appropriate color in color-buffer. Polygons are rendered from left one to right, and if z values are equal the subsequent will be drawn. Initial value is 0 (white) and the view is oriented from positive z axe to the origin.

	1	2	3	4	5	6	7	
	1	2	3	4	5	6		
	1	2	3	4	5			
	1	2	3	4				
	1	2	3					
	1	2						
	1							

1								
1	2							
1	2	3						
1	2	3	4					
1	2	3	4	5				
1	2	3	4	[illegible]	6			
1	2	3	4	5	6	7		
1	2	3	4	5	6	7	8	
1	2	3	4	5	6	7	8	9

1								
1	2							
	2	3						
		3	4					
			4	5				
				5	6			
					6	7		
						7	8	
							8	

Depth-buffer

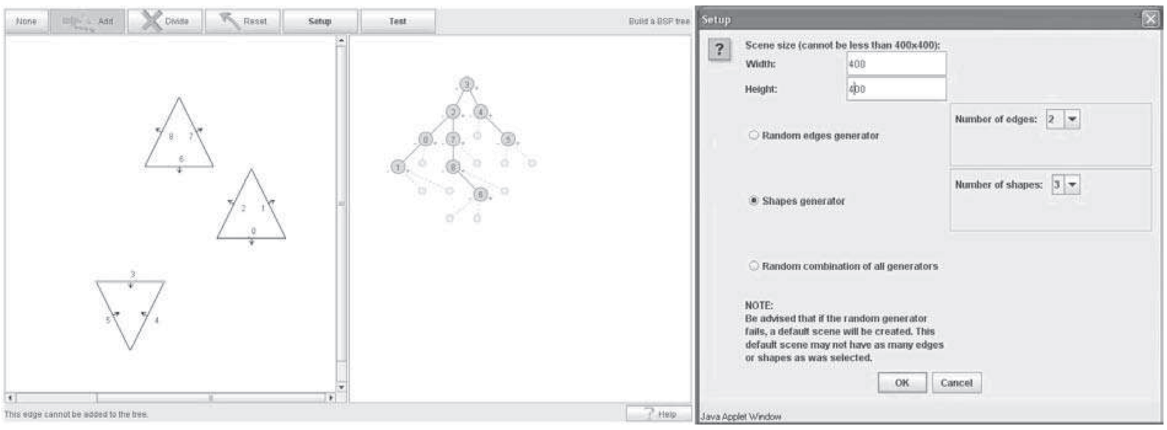
1	1	2	3	4	5	6	7	0
1	2	2	3	4	5	6	0	0
1	2	3	3	4	5	0	0	0
1	2	3	4	4	0	0	0	0
1	2	3	4	5	0	0	0	0
1	2	3	4	[illegible]	6	0	0	0
1	2	3	4	5	6	7	0	0
1	2	3	4	5	6	7	8	0
1	2	3	4	5	6	7	8	0

Color-buffer

▨	▨	▨	▨	▨	▨	▨	▨	
▨	▨	▨	▨	▨	▨	▨		
	▨	▨	▨	▨	▨			
		▨	▨	▨				
			▨	▨				
				▨	▨			
					▨	▨		
						▨	▨	
							▨	

Applet form_10_data_10 started

Fig. 6. Examples of computer graphics problems. Bresenham's line drawing algorithm is shown on the left. In each raster point that should be drawn an error value should be entered in the input box. Depth-buffer technique is shown on the right. In this problem three polygons with appropriate depth values are given. The goal is to determine the final value in the depth-buffer and appropriate colours in colour-buffer, after given polygons are drawn.



This edge cannot be added to the tree.

Scene size (cannot be less than 400x400):
Width: 400
Height: 400
○ Random edges generator
Number of edges: 2
● Shapes generator
Number of shapes: 3
○ Random combination of all generators

NOTE:
Be advised that if the random generator fails, a default scene will be created. This default scene may not have as many edges or shapes as was selected.

Java Applet Window

Fig. 7. Example of interactive BSP (Binary Space Partitioning) tree creation (left) with parameters for control of the BSP scene complexity (right).

'Add' function is active, each edge on the left is selected and placed in an appropriate node on the right, where a tree representation should be built. Node 3, in this example, has been selected as the root node. The first node on the positive side is node 4, while on the negative side node 2 is chosen.

If edge 7 is selected as the root node, then edges 0 and 2 should be divided. During creation of the scene, all possible intersections are marked with dots. Depending on the chosen order of the edges in the tree structure, a student divides appropriate edges. This example has been built using the StudTest framework. Another version of the same example has been created as a stand-alone applet [32].

Figure 7 right presents the relevant parameters in ProblemEditor for this example. To control the complexity of the scene, the desired number of edges or shapes (triangles, quadrilaterals) could be placed in the scene, or chosen by random selection. The number of edges ranges from two to nine and number of shapes from one to six. Based on given parameters for each student a new scene is created. Possible interactions of edges are controlled by the editor in order to keep the scene clear and obvious. Evaluation of solution accuracy for this problem depends on the relative position of each edge in the tree structure with respect to the other edges in the scene. Thus, partially correct solutions can also be evaluated.

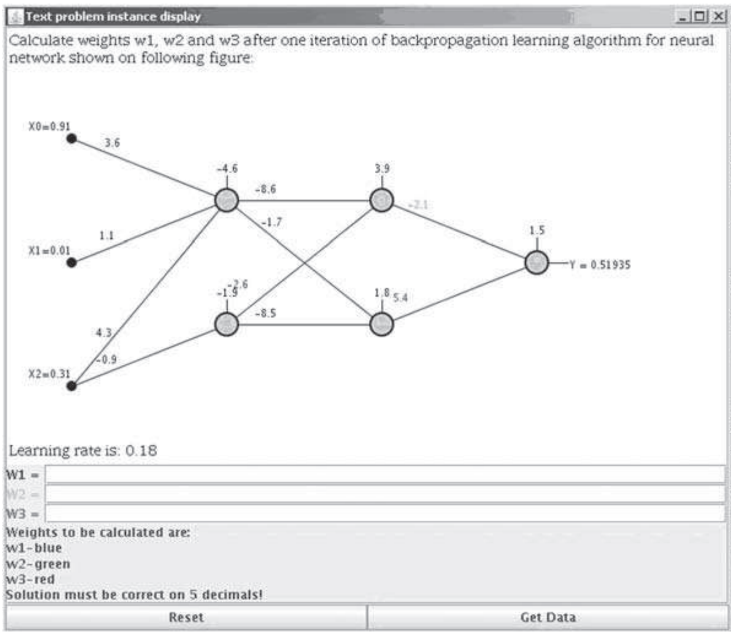
Another example with BSP trees is where the scene, appropriate tree and camera position are given in advance, but the node that contains the camera position must be determined. For this problem, the correctness of the solution can only take a binary response: correct or not correct. A third task is to generate a rendering order from the furthest edge to the closest edge for a given scene, BSP tree and camera position. This task is usually called BTF: Back to Front sorting (http://www.zemris.fer.hr/predmeti/rg/seminari/07_Prokopec/index.html /Examples/Building a Rendering Order).

5.4 Implementation of artificial intelligence problems

We present two examples of problems implemented for a course on artificial intelligence: neural networks and fuzzy control systems. Both of these problems deal with soft-computing; however, we have implemented a number of other problems covering propositional logic, predicate logic, Bayesian classifiers, and so on. We show screenshots of both problems from Problem Development Framework, a set of tools accompanying the StudTest framework that enable offline problem development and testing before actual deployment in StudTest.

1) Learning a neural network based on the back-propagation algorithm

Neural networks [33] are a fundamental soft-computing model that can solve hard problems such as nonlinear prediction and pattern recognition. Neural networks are composed from many smaller and simpler processing elements called artificial neurons, which are then interconnected. There are many types of neural network architectures, and one of the most well understood is that of feed-forward artificial neural networks. Knowledge in such a network is distributed across the neuron interconnections, modelled as weights. Correct weights are learned by the network

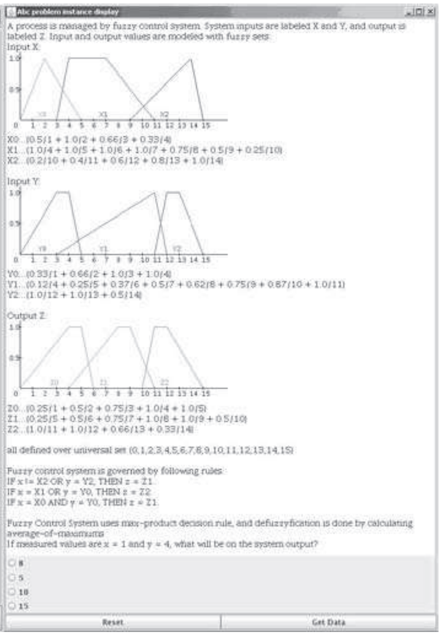


Calculate weights w1, w2 and w3 after one iteration of backpropagation learning algorithm for neural network shown on following figure:

Learning rate is: 0.18

W1 =
W2 =
W3 =

Weights to be calculated are:
w1-blue
w2-green
w3-red
Solution must be correct on 5 decimals!



A process is managed by fuzzy control system. System inputs are labeled X and Y, and output is labeled Z. Input and output values are modeled with fuzzy sets:

X0...(0.5/1 + 1.0/2 + 0.66/3 + 0.33/4)
X1...(1.0/4 + 1.0/5 + 1.0/6 + 1.0/7 + 0.75/8 + 0.5/9 + 0.25/10)
X2...(0.2/10 + 0.4/11 + 0.6/12 + 0.8/13 + 1.0/14)

Y0...(0.33/1 + 0.66/2 + 1.0/3 + 1.0/4)
Y1...(0.12/4 + 0.25/5 + 0.37/6 + 0.5/7 + 0.62/8 + 0.75/9 + 0.87/10 + 1.0/11)
Y2...(1.0/12 + 1.0/13 + 0.5/14)

Z0...(0.25/1 + 0.5/2 + 0.75/3 + 1.0/4 + 1.0/5)
Z1...(0.25/5 + 0.5/6 + 0.75/7 + 1.0/8 + 1.0/9 + 0.5/10)
Z2...(1.0/11 + 1.0/12 + 0.66/13 + 0.33/14)

all defined over universal set (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)

Fuzzy control system is governed by following rules:
IF x != X2 OR y = Y2, THEN z = Z1
IF x = X1 OR y = Y0, THEN z = Z2
IF x = X0 AND y = Y0, THEN z = Z1

Fuzzy Control System uses max-product decision rule, and defuzzyfication is done by calculating average-of-maximums
If measured values are x = 1 and y = 4, what will be on the system output?

○ 8
○ 5
○ 10
○ 15

Fig. 8. Examples of artificial intelligence problems: learning a neural network with back-propagation algorithm (left), fuzzy control system (right).

during the learning phase; back-propagation is one of the most frequently used weight-learning algorithms. The developed problem is shown in Fig. 8 left. During problem instantiation, a network architecture is randomly created (with feed-forwardness enforced) and initial weights are randomized. Then, the student is asked to perform one step of the back-propagation algorithm for the given network input and requested output, and to calculate new weights. A helper object renders the created network graphically. As we can see, each student gets a different neural network with different set of inputs and outputs, and must solve the problem alone.

2) Fuzzy control systems

Fuzzy control systems [34] are widely used to control consumer electronics. They can work with imprecise data and are based on IF-THEN rules. They are grounded in the theory of fuzzy sets and fuzzy relations. Figure 8. right shows an example of a fuzzy control problem. A student is presented with a fuzzy control system having two inputs (*X* and *Y*) and a single output (*Z*). Input values *X* and *Y*, and output value *Z* are linguistic variables, with terms *X0*, *X1*, *X2*, *Y0*, *Y1*, *Y2* serving as inputs and *Z0*, *Z1*, *Z2* as outputs. An appropriate fuzzy set models each of these terms and is displayed both graphically and textually. Fuzzy IF-THEN rules are then defined. Students are asked to determine the system output for a given input. In this problem, both the input terms and associated fuzzy sets and the output terms and associated fuzzy sets are generated randomly. IF-THEN rules are also created randomly, as is the system input value, so that different students get different problems to solve. In order to evaluate this kind of problem, the evaluator must implement a fuzzy control system programmatically, which can easily determine the correct answer. Helpers render the created fuzzy sets and linguistic variables graphically.

5.5 Implementation of digital logic problems

We present two sample problems implemented for a Digital Logic course: determining the Boolean function for a given CMOS implementation, and realizing the Boolean function in PLA circuits. These represent a small but illustrative subset of the problems developed to cover the material in this course.

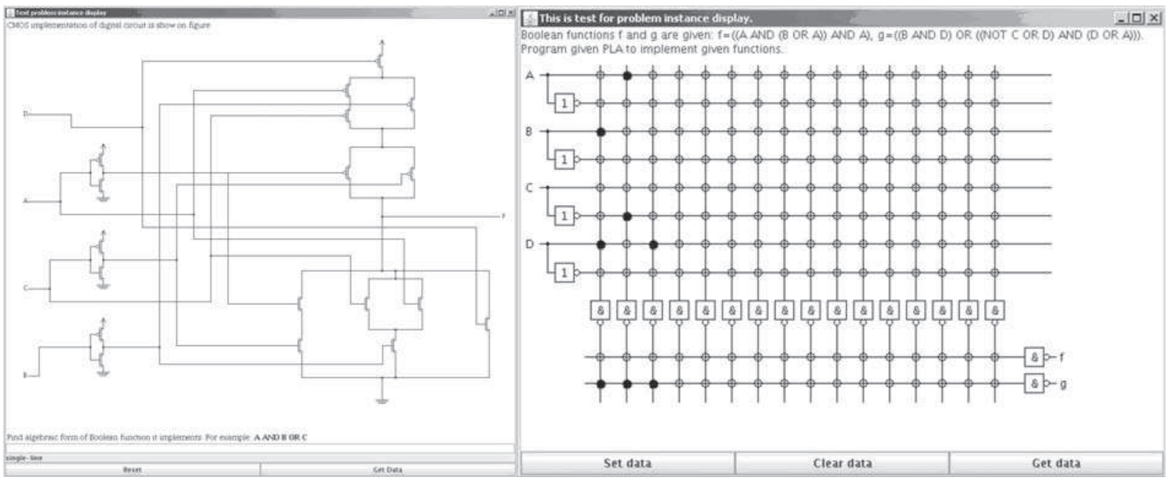
1) Determining the Boolean function for a given CMOS implementation

Boolean functions are often implemented in CMOS since this technology has many satisfactory properties, with the most significant being low power consumption [35]. Students learn about CMOS technology in a digital logic course. The first problem we model creates random Boolean functions and asks students to draw their implementations [28]. The circuit is checked via simulation. The simulator developed to evaluate this problem is also publicly available online [36], so that students can experiment with various digital CMOS circuits.

Figure 9 left shows the second problem. For each student, the problem instantiator constructs a random Boolean function, implements it in CMOS, and then presents the CMOS schema to the student. The student must then analyse this circuit and determine which function it implements. In order to solve this problem correctly, the student must know how Boolean functions are implemented in CMOS, and enter the algebraic form of the Boolean function. The evaluator will then check if the solution is correct by comparing the logical output of the original function to the student's solution. The helper renders the CMOS schema graphically.

2) Realization of Boolean functions in PLA circuits

The PLA circuit is one of programmable digital circuits [37] taught in the digital logic course (along


Test problem instance display

CMOS implementation of digital circuit is show on figure

Find algebraic form of Boolean function it implements. For example: A AND B OR C

This is test for problem instance display.

Boolean functions f and g are given: f=((A AND (B OR A)) AND A), g=((B AND D) OR ((NOT C OR D) AND (D OR A))).

Program given PLA to implement given functions.

Fig. 9. Examples of digital logic problems: CMOS implementation of Boolean function (left), programming of PLA (right).

Table 1. Summary of the use of StudTest framework in five courses

Course, academic year	Number of students	Number of homework assignments	Number of problems per assignment	Total number of problems
DL, 2005–2006	1103	3	15	50
DL, 2006–2007	948	3	20	70
DL, 2007–2008	834	5	10	90
CG, 2006–2007	58	1	10	20
CG, 2007–2008 a	65	2	8	25
ICG, 2007–2008 b	101	2	8	25
AI, 2007–2008	94	1	10	10

with PAL-s and FPGA-s). In order to help students to better learn how these circuits work, we have developed a problem as shown in Fig. 9 right. In this problem, the student is presented with a PLA of one of the following types: AND-OR, OR-AND, NAND-NAND, NOR-NOR. The problem instantiator randomly creates the algebraic form of a Boolean function that the student must implement. This problem is presented to the student with a Java applet, which allows the student to solve the problem by clicking on programmable interconnection switches. Figure 9 right shows a partially programmed PLA for a function *g*. Once solved, the problem evaluator will simulate the PLA circuit programmed by the student in order to verify that it was correctly programmed. Note that special care must be taken with this kind of problem, since there can be hundreds or thousands of equally correct solutions, so simple comparison with the expected solution is not appropriate.

6. EVALUATION

For evaluation, the hypotheses we wanted to test were that the proposed architecture provides a reusable, extensible, robust framework for web-based assessment that significantly improves students' knowledge in an attractive and qualitative way. The proposed framework has been used since 2005 to support several courses: digital logic, computer graphics, interactive computer graphics and lately artificial intelligence.

The framework was first tested on homework assignments for a course in digital logic in 2005–06. The course included three assignments with approximately fifteen problems each. Table 1 summarizes the number of students, homework assignments, problems per assignment and total number of problems. The first column identifies a course as digital logic (DL), computer graphics (CG), interactive computer graphics (ICG) and artificial intelligence (AL) and its associated academic year. The second column presents the number of students in each generation. In the academic year 2005/06 there were a considerable number of students in DL who did not pass the course in a previous year, so the total number of students was very high. During the next few years, the number of students stabilized. The transition period corresponded with the start of the Bologna process at our University in 2005.

During the first DL homework assignment in 2005–06, the allotted period for the assignment was one week and the deadline for finishing the assignment was fixed. Most students waited until the last minute to press the 'Finish' button and send their results. Because of simultaneous access by a large number of students, there was a case of data overload and the whole system collapsed. When the system was restored, the deadline was extended to avoid complaints from angry students. All the results were correctly uploaded and evaluated. The main drawbacks of the system were exposed and repaired. After the initial period, the system worked well. The last column in Table 1 present the total number of problems in the pool of all problems, which grows each year.

The proposed framework supports development of various question types and problems, including interactive ones such as BSP tree building. Our framework is extensible and reusable because the core part of the software architecture is the same for all courses; only the prlets differ for each problem, as presented in Fig. 2. Reusability is also exhibited by the fact that the same problems can be used to produce individualized assessments for a large class and for different semesters. This is because they are parameterized. The whole system appears to be very robust, since several thousands of students can use the system to solve their assignments. Hence, peak overload is handled properly.

The total number of problems in the system library grows each year, so the probability that two students will get the same problem types continually decreases. Over time, students noticed that there was no way to cheat on the assignments. Moreover, instructors strongly encouraged the students to learn, practice and understand each problem type because similar ones would appear in the midterm or final exam. It is important to convince the students of that fact, because it provides additional motivation for them to solve the given assignments by themselves, beginning with the second homework. We noticed that some of them chose to work in groups, not to cheat, but rather to help each other and to learn how to solve different problem types. Therefore, our goal of motivating the students to learn and improve their knowledge was achieved.

The incentive for students in the digital logic course was 15 credit points (15%). Students said that they preferred more distributed assessments in the semester with fewer problems, so we replaced three homework assignments of twenty problems each with five homework assignments of fifteen problems each. We also changed successive access to problems with direct access to each problem. A similar experience was reported in [26].

In CG in 2006–07, homework was optional, but five extra credit points (5%) were offered for completing it. In CG in 2007–08, two groups were studied, one given an optional five extra credit points (CG 2007–08), and one given five credit points for the course ICG 2007–08 (5%). Students suggested increasing the number of credit points because of the required time to solve the homework. In the AI course, students who solved homework correctly were given 1.67 credit points. In Table 2, we can see that the number of students who solved more than one homework assignment (column '1 HW') is very high, especially when more credit points were offered. The number of students who solved all proposed homework assignments is slightly smaller, but it is important to notice that the proportion of students who solved all homework assignments among students who passed the exam was especially high: almost all cases were around 90% or higher.

For an objective evaluation of StudTest, we observed the results of the homework assignments, two midterms and the final exam. The final course grade was a combination of homework assignments, three exams, laboratory work, and classroom activities. Students also completed entry, midterm and final questionnaires during the semester that were very useful for course review. From those data we calculated correlations. To show that a prediction of exam results can be made based on homework results, we used regression analysis to establish the relationship between these results.

Over the last three years, we collected a significant quantity of data from the use of StudTest in various courses. We evaluated the linear relationship between exam results and appropriate homework covering the same course material using Pearson's correlation (linear relationship model).

As exemplified in Table 3, in DL during 2005–2006, we evaluated the linear relationship between the first homework (HW1) and the first midterm exam, the second homework (HW2) and the second midterm exam (MI2) and finally between the third homework (HW3) and the final exam (ZI). In the regression analysis we included only those students who completed both the homework and the related exam in the number of observations. This is the reason why the number of students in the sample is less than the number of students in the course (due to illness, travel, etc). An analysis using Pearson's correlation coefficient indicated a statistically significant linear relationship between HW1 and MI1 (r = 0.394), HW2 and MI2 (r = 0.412), and HW3 and ZI (r = 0.341), assuming p=5%. We obtained similar results, all with statistically significant linear relationships, for all other courses and academic years.

Table 2. Statistics of the use of StudTest framework in homework assignments and corresponding exam results

Course, academic year	1 HW	all HW	Pass exam	(all HW & Pass exam)
DL, 2005–2006	1070 (97%)	973 (88%)	814 (74%)	797 (98%)
DL, 2006–2007	919 (97%)	793 (84%)	723 (76%)	698 (97%)
DL, 2007–2008	804 (96%)	626 (75%)	601 (72%)	540 (90%)
CG, 2006–2007	32 (55%)	32 (55%)	27 (47%)	26 (96%)
CG, 2007–2008	37 (57%)	21 (32%)	29 (45%)	20 (69%)
ICG, 2007–2008	88 (87%)	78 (77%)	83 (82%)	75 (90%)
AI, 2007–2008	73 (78%)	73 (78%)	77 (82%)	67 (87%)

Table 3. Regression analysis. Exam results and corresponding activity in homework assignments

					intercept			predictor variable		
Course	Predict.	Num. of observ.	R	R square	coef.	t stat	p value	coef.	t stat	p value
DL0506	HW1-MI1	1022	0.394	0.155	3.330	5.537	3.91E-08	0.105	13.689	2.76E-39
	HW2-MI2	1031	0.412	0.170	3.360	5.879	5.56E-09	0.103	14.493	1.86E-43
	HW3-ZI	876	0.341	0.116	7.517	12.256	5.55E-32	0.090	10.717	2.88E-25
DL0607	HW1-MI1	894	0.402	0.161	2.460	5.172	2.85E-07	1.601	13.094	5.84E-36
	HW2-MI2	834	0.371	0.138	4.228	9.586	1.02E-20	1.445	11.525	1.28E-28
	HW3-ZI	779	0.336	0.113	5.734	6.963	7.07E-12	2.265	9.960	4.48E-22
DL0708	HW1-MI1	766	0.230	0.053	4.487	7.317	6.42E-13	1.587	6.544	1.09E-10
	HW2,3-MI2	713	0.323	0.104	5.227	6.763	2.81E-11	1.557	9.103	8.7E-19
	HW4,5-ZI	647	0.364	0.133	3.746	4.048	5.8E-05	2.009	9.938	9.51E-22
CG0607	HW-KZ2	27	0.473	0.224	10.814	7.013	2.37E-07	1.099	2.686	0.012665
CG0708	HW2-ZI	19	0.476	0.226	10.128	2.559	0.020329	13.981	2.231	0.03946
ICG0708	HW2-ZI	78	0.365	0.133	7.685	5.977	6.88E-08	5.842	3.419	0.001011
	HW1-MI1	85	0.295	0.087	5.949	2.832	0.005798	7.512	2.815	0.006095
AI0708	HW3-ZI	65	0.265	0.070	6.712	2.171	0.033684	1.116	2.178	0.033194

It is important here to mention that showing a statistically significant linear relationship cannot be interpreted as proving cause and effect. However, students often told us that doing the homework was a good way to prepare themselves for exams and to try to solve many complex and difficult problems. Also, some students who could not solve the homework alone sought help, further contributing to their improved results on both homework and exams.

This clearly shows the benefits of including such a system in a course: since students get their own versions of the problems, they must solve the problems themselves. If a student solves it incorrectly, the system can be configured to give the student another newly-created instance of the problem, letting the student learn and try again, eventually resulting in better student knowledge.

7. NEW ISSUES IN THE LEARNING PROCESS

Developing the framework for the system was very demanding. Many students contributed to the development of the StudTest framework, not only by providing programming expertise but also with their ideas and creativity. Experiences gained through this project, for students and instructors, were invaluable. Based on the StudTest framework, problems for several courses were implemented and tested on several generations of students. In each new generation, the system was improved and fresh ideas for the problem tasks appeared. Student participation is always very important in the definition of new problems, because students have a different perspective on a specific task from the instructors. From the student's point of view, some particular problems are difficult to understand and clarify: thus, it is very important to define tasks such that solving is directed toward understanding and learning.

Another important advantage of the proposed system is that it makes the learning process attractive, since students must continually refocus their attention each time a new scene is created until they achieve full understanding. The auto-evaluation component is crucial for asynchronous communication, allowing the learning to occur on the student's timetable. Multi-technology is supported, so students can learn wherever and on whatever platform they choose. Moreover, each problem inspires the creation of a new one. Students retained high enthusiasm and passion, and influenced the continuous improvement of course materials, not only in quality of the materials but also in terms of new vision and challenges for learning and assessment methods.

8. CONCLUSIONS

Currently, there are many attempts to make the learning process better, more interesting and more interactive. People are constantly changing, and today, we—the digital immigrants—are trying to visualise learning techniques for the generation of digital natives. In this new world, students are learning from digital materials; they learn by playing games, by participating in virtual worlds and by performing experiments in virtual and/or remote laboratories or simulators. An important part of this picture is problem-based learning, in which learning is centred around problems that must be understood and solved. Problems can be, and often are, integrated in games, however, usually very specialized ones. For instance, games bundled with an excellent physics simulator are useless for learning of quantum theory or optics.

We have demonstrated another approach, which offers a context-free foundation for building problems and performing assessments. Supported problems can be quite simple, or very complex. They can have a simple user interface, or rich graphical interface. They can be instantiated and evaluated by provided Java-based programs; for those purposes they can even utilize arbitrary additional software tools available on site (such as Wolfram's Mathematica, MatLab, etc.). This kind of platform can be utilized in many ways to enhance the learning process or assess student knowledge.

In its simplest form, StudTest can be used as an assessment tool. Given the problems, StudTest can generate individualized questions for each student, effectively preventing cheating. Given the fact that problems support self-evaluation, results can be made available to students immediately, providing rapid feedback, which is appreciated by students.

StudTest can also be used as a tool for student self-evaluation. Given the problems, teachers can set up a number of small topic-focused tests, which students can use to obtain an objective evaluation of their topic-specific knowledge. This is particularly important when preparing for formal exams. These tests can be accessed multiple times; each time generated problems will be different, which is important to maintain students' interest.

Finally, students can learn through the creation of new problems. To foster the development of new problems, we have created a Problem Development Framework, which is an environment suitable for rapid problem creation. Each year we provide an opportunity for certain students to try and create a new problem from a given topic. Being in such a situation, the students become highly motivated to learn the topic, and to obtain a deep understanding, since this is necessary for a successful problem development, where questions such as 'what if this happens, and what if that happens' occur constantly. Added value is that developed problems remain in a problem database, enabling progressively better knowledge assessment and self-evaluation.

Feedback obtained from students is also very positive. Many of them very much appreciate the fact that they can perform a self-evaluation at their own time and pace, and that they get an immediate feedback. Students involved in problem creation may even be more satisfied, since they mastered the topic. They were motivated and interested, and felt that they have made a contribution to the following generations of students.

Most importantly, StudTest is not a specific system, but supports a wide range of applications, which we demonstrated on three very different university-level courses. In the future, we expect that a number of involved courses will also become larger. A repository of developed problems could also be established to enable others to benefit from problems that are already developed. We believe that a described system for knowledge assessment and training, which can work with complex problems, is beneficial both for educators and for students.

Acknowledgments—We would like to thank the many students who participated in the realization of the basic framework for their very enthusiastic involvement in this project. This work has been carried out within projects 036-0362980-1921 Computing Environments for Ubiquitous Distributed Systems and 036-0361994-1995 Universal Middleware Platform for e-learning Systems funded by the Ministry of Science, Education and Sport of the Republic of Croatia.

REFERENCES

1. Moodle, Available at http://www.moodle.org (Accessed September 2008).
2. The Blackboard Learning System, http://www.blackboard.com (Accessed September 2008).
3. V. Glavinić, M. Čupić, S. Groš, S, Nescume, A system for managing student assignments, In *Proceedings of the First International Conference on Internet Technologies and Applications (ITA 05)*, Wrexham, North Wales, UK, 2005, pp. 233–238.
4. P. Wallace, Blending instructional design principles with computer game design: The development of Descartes' cove, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2005*, Ed-Media, Montreal, Canada, 2005, pp. 402–407.
5. C. C. Ko, Ben M. Cheng, J. Chen, J. Zhang and K. C. Tan, A Web-based Laboratory on Control of a Two-Degrees-of-Freedom Helicopter, *Int. J. Eng. Educ.,* **21**(6), 2005, pp. 1017–1030.
6. W. L. Chan, Z. Qu, Using XML/Java to Enhance an Online Learning Architecture for Engineering Education, *Int. J. Eng. Educ.*, **21**(2), 2005, pp. 288–296.
7. H. Wang, Performing a course material enhancement process with asynchronous interactive online system, *Computers & Education*, **48**(4), 2007, pp. 567–581. 10.1016/j.compedu.2005.03.007
8. B. A. Foss and T. I. Eikaas, Game Play in Engineering Education Concept and Experimental Results, *Int. J. Eng. Educ.*, **22**(5), 2006, pp. 1043–1052.
9. IEEE Standard for Learning Object Metadata, 2002, 1484.12.1, Available at http://ltsc.ieee.org/wg12
10. SCORM—Sharable Content Object Reference Model specification, (2004), Available from: http://www.adlnet.gov
11. IMS Global Learning Consortium Inc, IMS specifications, Available from: http://www.imsglobal.org/specifications.html
12. IEEE PAPI. IEEE P1484.2.5/D8, 2002. Draft Standard for Learning Technology—Public and Private Information (PAPI) for Learners (PAPI Learner).
13. M. Kravcik, M. Specht, Authoring Adaptive Courses—ALE Approach. *Advanced Technology for Learning*, **1**(4), 2004, 215–220.
14. P. Dolog, N. Henze, W. Nejdl, M. Sintek, The personal reader: Personalizing and enriching learning resources using semantic web technologies, In *Proceedings of the Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH)*, Eindhoven, Netherlands, 2004, pp. 85–94.
15. P. Dolog, B. Simon, W. Nejdl, T. Klobučar, Personalizing access to learning networks. *ACM Transactions on internet technology*. 8(2, Art 8), 2008, 10.1145/1323651.1323654 ..
16. L. Aroyo, P. Dolog, G-J. Houben, M. Kravčík, A. Naeve, M. Nilsson, F. Wild. Interoperability in Personalized Adaptive Learning, *Educ. Tech. Soc.* **9**(2), 2006, pp. 4–18.
17. M. Kravčík, and D. Gašević, Adaptive hypermedia for the semantic web, In *Proceedings of the Joint international Workshop on Adaptivity, Personalization &Amp; the Semantic Web,* (Odense, Denmark, APS '06. ACM, New York, NY, 2006. pp. 3–10. http://doi.acm.org/10.1145/1149933.1149935.
18. M. Kravčík, & D. Gaševic, Leveraging the Semantic Web for Adaptive Education, *J. Interactive Media in Educ.*, (Adaptation and IMS Learning Design. Special Issue, ed. D. Burgos), (2007), http://jime.open.ac.uk/2007/06/
19. D. Dagger, A. O'Connor, S. Lawless, E. Walsh, V. Wade, Service oriented eLearning platforms: from monolithic systems to flexible services, *IEEE Internet Computing Special Issue on Distance Learning*, doi: 10.1109/MIC, 2007, p. 70.
20. C. Feier, A. Polleres, R. Dumitru, J. Domingue, M. Stollberg, D. Fensel, Towards intelligent web services: The web service modeling ontology (WSMO), *Int. Conf. Intelligent Computing (ICIC)*. 2005.
21. M. Kravčík, M. Specht, R. Oppermann, Evaluation of WINDS Authoring Environment, In: De Bra, P. & Nejdl, W. (Eds) *Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer, 2004, pp. 166–175.
22. D. N. Batanov, N. J. Dimmitt, & W. Chookittikul, Q&A teaching/learning model as a new basis for developing educational software, In *Proceedings of the 30th Annual Frontiers in Education –* Vol. 01, October 18–21, 2000, F2B.12-18, doi: 10.1046/j.0266-4909.2002.04800.x
23. G. J. Hwang, P. Y. Yin, S. H. Yeh, A tabu search approach to generating test sheets for multiple assessment criteria, *IEEE Transact. Educ.*, **49**(1), 2006, pp. 88–97.

24. J. Tvarožek, M. Kravčík, & M. Bieliková, Towards Computerized Adaptive Assessment Based on Structured Tasks, In Nejdl, W. et al. (Eds) *Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer Berlin / Heidelberg, 2008, 224–234.
25. P. Brusilovsky, and S. Sosnovsky, Engaging students to work with self-assessment questions: A study of two approaches, In: *Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE'2005, Monte de Caparica, Portugal), 2005, pp. 251–255.
26. P. Brusilovsky and S. Sosnovsky, Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK, *J. Educ. Resour.*, Comput. **5**(3), 2005, art. 6, DOI http://doi.acm.org/10.1145/1163405.1163411.
27. S. Pathak and P. Brusilovsky, Assessing Student Programming Knowledge with Web-based Dynamic Parameterized Quizzes. In: Barker, P. and Rebelsky, S. (Eds*) Proc. of ED-MEDIA'2002—World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Denver, CO, June 24–29, 2002, pp. 1548–1553.
28. V. Glavinić, M. Čupić, S. Groš, StudTest—a platform supporting complex and interactive knowledge assessment, *International Conference ICL—Interactive Computer Aided Learning,* Villach, 2008.
29. Problettes—The Home Page http://www.problets.org/
30. G. Booch, Object-oriented analysis and design with applications, 2nd ed, Addison Wesley, 1994.
31. D. Hearn and M. P. Baker, Computer Graphics with OpenGL, 3rd ed, Prentice Hall, 2003.
32. A. Prokopec, Z. Mihajlovic, Binary Space Partitioning Applet, Interactive applet for learning how to create and use BSP trees, Available at: http://www.zemris.fer.hr/predmeti/rg/seminari/07_Prokopec/index.html, 2007.
33. P. Picton, *Neural Network*, Palgrave Macmillan, 2Rev Ed edition, 2000.
34. H.-J. Zimmermann, *Fuzzy Set Theory—and Its Applications*, Kluwer Academic Publishers, 2nd ed, 1991.
35. D. D. Gajski, *Principles of Digital Design*, Prentice Hall, 1997.
36. M. Čupić*, Interactive Digital Circuit Simulator for CMOS Digital Circuits and Logical Gates,* Available at http://www.zemris.fer.hr/predmeti/de/Appleti/applet/applet.html (Accessed November 2008).
37. S. Brown, Z. Vranesic, *Fundamentals of Digital Logic With VHDL Design* McGraw-Hill, 2000.

Marko Čupić received the B.S. degree in Computer Science in 2002 and the M.S. degree in Computer Science in 2006 from the Faculty of Electrical Engineering and Computing, University of Zagreb. He is currently a Researcher at the Department of Electronics, Microelectronics, Computer and Intelligent Systems, of the University of Zagreb, Croatia. His current research interests include soft computation and e-learning.

Željka Mihajlović received the B.S. degree in Electrical Engineering in 1988, the M.S. and Ph.D. degrees in Computer Science in 1993 and 1998 respectively, from the Faculty of Electrical Engineering and Computing, University of Zagreb. She is currently an Associate Professor at the Department of Electronics, Microelectronics, Computer and Intelligent Systems, of the University of Zagreb, Croatia. Her current research interests include computer graphics and visualization algorithms, reconstruction techniques as well as e-learning technologies.