Coping with Plagiarism and Grading Load: Randomized Programming Assignments and Reflective Grading[1,2]

Diomidis Spinellis, Panagiotis Zaharias and Adam Vrechopoulos

Department of Management Science and Technology

Athens University of Economics and Business

Abstract

Programming assignments often suffer from plagiarism and lack of feedback. The Jarpeb system creates individually randomized assignments, grades the students' programs by utilizing Java's reflective evaluation capabilities, and allows students to submit their grade through the web by signing their grade with a cryptographically strong checksum. Jarpeb's empirical evaluation included as the dependent variables important learners' dimensions: plagiarism, understanding, learning, fairness, difficulty, fun, and interest. The results indicate that Jarpeb contributes to the reduction of plagiarism, increases the understanding and learning of the course subject while also increasing the perceived fairness, fun and interest of the learners. The system, however, proved to increase the difficulty of the related exercises. We discuss the implications for educators and outline specific future research directions.

Keywords: Plagiarism, grading, programming, reflection, Java

Introduction

Programming assignments in computer science, information technology, and related courses often suffer from plagiarism and lack of feedback. Both can be directly attributed to the ratio between instructors and students. Instructors and lab assistants have minimal time to examine each programming coursework, ensuring that it is correctly implemented, and that it is not a copy of the work of another student. Unfortunately, the reality of an increasing ratio of students to instructors in many universities throughout the world makes the obvious solution—increased instructor time to work with the students—difficult to implement. Some schools have resorted to plagiarism detection tools and automated marking systems, while others minimize the importance of programming assignments in a student's final grade.

During the 2003–2004 and 2004-2005 winter semesters we tried a different approach for our Object-Oriented Programming course. Students could download from the course's web page Jarpeb: a system that hands out and grades randomized programming assignments. Jarpeb uses a number of programming techniques, such as reflection, cryptographic checksums, and inheritance

---

[1] Diomidis Spinellis, Panagiotis Zaharias and Adam Vrechopoulos. Coping with plagiarism and grading load: Randomized programming assignments and reflective grading. *Computer Applications in Engineering Education*, 15(2):113–123.

[2] This is a machine-readable rendering of a working paper draft that led to a publication. The publication should always be cited in preference to this draft using the reference in the previous footnote. This material is presented to ensure timely dissemination of scholarly and technical work.

to provide a level of functionality not found in earlier systems. In the following sections we present the Jarpeb system, and the results we obtained by testing a number of hypotheses we derived through the literature review. The objective of the paper is, therefore, dual: on the one hand, the paper presents the design of an e-learning system realized through a number of innovative implementation techniques, while on the other the paper uses the system as a vehicle to test specific research hypotheses provided through the emerging e-learning literature. Before moving on with the description of the system, let us first examine the main pedagogical considerations, on which the development of the system is based.

Pedagogical concerns associated with instructional assessment in e-learning environments

The development of the Jarpeb system falls under the area of assessing programming assignments. We view assessment as an integral part of the teaching and learning process, rather than an afterthought. Assessment is the process of gathering, describing and/or quantifying information about learner performance (Rovai, 2000). Assessing computer science topics and especially computer programming is a hard and demanding task. In industry, situations where each (experienced) programmer has one tester at her disposal are not unusual, yet in an educational environment we often find one teaching assistant grading the work of tens or hundreds of (novice) programming students. The pedagogical theory behind science education relies heavily on the contemporary theories and techniques of learning such as constructivism (Hazzan and Lapidot, 2004; Ben-Ari, 2001) and active learning (Newman *et al*, 2003; McConnell, 1996). The wide adoption and implementation of constructivist and active learning paradigms has also affected the notion of assessment and its strategies. Traditional assessments follow a selected response strategy (Rovai, 2000) where learners choose a single answer from a given set of response items, in the form of multiple choice, true-false, matching, and other alternatives. Most of the current e-learning developments follow this traditional interpretation of assessment, thus making assessment a relatively weak component of most e-learning systems and technologies (Schank, 2001). Vendors, instructional designers, and subject matter experts alike appear to struggle with conceiving assessment as anything more than a multiple-choice test that usually measures low-level cognitive learning objectives, such as knowledge and comprehension (Reeves and Aggen, 2002).

Consequently there are many full fledged Learning Management Systems and Web course authoring tools available that integrate Web-based testing and assessment capabilities by providing templates for multiple-choice questions, true/false questions, matching questions, or short answer questions. However, these tools omit essay questions, programming projects, lengthy assignments, and case studies. For example, assessing learning in topics such as software development and programming is a challenging task that is associated to higher-level learning objectives and cannot be effectively accomplished through multiple-choice questions and the like. The development of the Jarpeb system has as its main goal to assess programming assignments, supporting the students in an iterative process of learning assessment. While developing the Jarpeb system we tried to address several issues that are directly associated with the assessment process. To that end, we took into account a number of crucial factors intimately related to assessment such as: plagiarism (Howard, 1995), instructional feedback, perceived fairness and difficulty, as well as the learners' affective responses (Worthman, 1999; Goleman, 1995).

Plagiarism

Cheating with programming assignments is an old bad habit of young programmers and students. Authors use the term "plagiarism" as an umbrella term for cheating, nonattribution and patchwriting; See (Howard, 1995) for the exact definitions. Plagiarism has always been a major problem in the educational process. It has been found that factors which increase plagiarism include poor understanding of what constitutes plagiarism, leniency of instructors regarding

plagiarism, pressures related to grades, a large number of assignments, lack of time, wanting to avoid hard work, and little interest in the topic. In contrast, factors that appear to reduce plagiarism include positive emphasis professional ethics, fear and guilt, personal confidence, and a desire to learn (Love and Simmons, 1997; Schneider, 1999). Regarding the latter, a main objective for developing the Jarpeb system was not to evoke feelings such as fear and guilt but to increase the learners' personal confidence and interest on the topic of the assignment, thus reducing plagiarism. Therefore, we formulated the following research hypothesis:

H1: The Jarpeb system significantly contributes to the reduction of plagiarism

Instructional feedback

Feedback is another main concept of our implementation. Feedback is a vital and indispensable instructional activity (Aronson and Briggs, 1983) during which misconceptions are corrected. Providing feedback is crucial for the orientation and motivation of the learner (Spitzer, 1996; Driscoll, 2002). Effective e-learning systems can provide rich and meaningful feedback to develop the learners' ability to self-diagnose their learning problems and correct errors without prompts. Additionally, inadequate feedback is one of the main reasons associated with the learners' frustration (Driscoll, 2002). Feedback as provided by Jarpeb system should assist learners in *learning* the subject matter (ie programming using the Java language) and *understanding* the exercise code of their assignments.

Learning cannot be approached as a conventional task, as though it were just another kind of work, with a number of problems to be solved and various outputs to be produced (Mayes and Fowler, 1999). There are important differences between the typical software and web users and the users as learners. When the users are learners the emphasis must be put on interfaces that support "learning while doing tasks" rather than those interfaces that support "doing tasks" (Hsi and Soloway, 1998).

Understanding is widely recognized as an important outcome of education (Perkins and Unger, 1999). Learning with understanding may yield higher engagement as well as more active use and transfer of knowledge (Perkins and Unger, 1999). In addition thinking and deep understanding is crucial when learning how to program (Cernuda del Rio, 2004). For example debugging is a demanding mental process that requires a lot of attention, critical thinking and deep understanding of the exercise code. Hence every learning environment or system or technology that assist learners to learn programming should foster understanding of the exercise code.

Elaborating on the aforementioned feedback-related dimensions, we formulated the following research hypotheses:

H2: The Jarpeb system significantly increases the *understanding* of the exercise code

H3: The Jarpeb system significantly contributes to *learning* the subject matter

Perceived fairness and difficulty

Perceived fairness and difficulty are intimately related to the assessment process either conducted in a traditional educational setting or with the use of information and communication technologies. When the perceived difficulty of an assignment or exercise is high, the assignment can cause negative emotional responses and frustration to the learners. In addition, the perceived fairness is a crucial issue in every assessment process, especially those delivered through computer-based applications. Several studies (Duffield and Spencer, 2002; Olson *et al*, 2000) researching perceived fairness of an assessment process have shown that marking criteria (as well as positive or negative marking), coverage of the learning material, ambiguity of questions and phrases used in the assessment material, and relevance of assessment with the knowledge and

skills to be taught, are important determinants of perceived fairness. Our corresponding research hypotheses regarding Jarpeb are:

H4: The Jarpeb system significantly increases the perceived *fairness*

H5: The Jarpeb system significantly affects the perceived *difficulty* of the exercises

Affective considerations

The increasing importance of the affective dimension

A lot of recent research emphasizes the interplay of emotions and learning (Picard *et al*, 2001; Konradt and Sulz, 2001; O'Regan, 2003). Emotions and affect have been shown to be significant in relation to attention, memory and decision making, all of which are of critical importance in the learning process (Worthman, 1999; Ferro, 1995). Affect goes beyond simple enjoyment; it plays an important role in the development of persistence and deep interest in a subject (Goleman, 1995). Importance of affect has also been emphasized by the latest advancements in Human Computer Interaction research (Norman 2004; Hudlicka, 2003; Partala and Surakka, 2003). For a description of the cognitive complexity of the programming process see the classic article by Cant *et al* (1995). It is critical that systems designers assess the range of possible affective states that users may experience while interacting with the system. Such an understanding provides input for possible effects that the system may have on the users and on task performance. Picard *et al* (2001) have conducted remarkable research work exploring the role of affect in science, math, engineering, and technology learning and the development of practical tools "*that can specifically aid the student learn how to learn*".

Focus on perceived interest and perceived fun

Furthermore, designers should be aware that the affective dimension of a system influences how learners pursue goals, develop preferences, build confidence, persist in the face of difficulty, establish priorities, and care about learning. In practical terms the game is about how to engage learners in the learning process, help them love learning, enjoy challenges, connect with the subject matter, and persist when things get tough during the learning process. Importantly, the staying power of negative affect tends to outweigh the more transient experience of positive affect. This is a phenomenon known as *negative asymmetry* (Giuseppe, Brass, 2003). Unfortunately, for the purposes of motivating learners this negative asymmetry means that negative affect experienced from failure will persist disproportionately to the positive affect experienced from success. Simply put, learning will be enhanced when negative emotions are minimized and positive emotions maximized. Therefore, an important goal is for developing an assessment process that will minimize the learners' irritation and frustration. Positive affect, such as the development of deep interest in the subject, strengthens learning promoting thinking that is efficient, but also careful, open-minded and thorough. Interest is one of the most basic cognitive emotions (Buck, 2002), but very little research work has been conducted to further investigate it in e-learning developments. The Jarpeb system has been developed according to these requirements with an explicit goal to increase the learners' interest and decrease their irritation when interacting with the system.

Perceived fun

As we already stated, designers should assess a range of possible affective states that users may experience while interacting with the system. While elaborating upon affective considerations and e-learning design, perceived fun emerges as one of the most important goals. Usually post-secondary and higher education are not fun undertakings but should learning be fun? There is an increasing interest concerning this matter (Okan, 2003; Bloom and Hanych, 2002) and as Rea

(1998) suggested fun and learning may co-exist in a negotiated balance. Fun encompasses pleasure and enjoyment. Such "*hedonic components*" (Hassenzahl, 2001) can greatly affect the users' experiences. Findings from technology acceptance studies (Mundorf *et al*, 1993; Igbaria *et al*, 1994) revealed that perceived fun had an even stronger effect on user satisfaction than perceived usefulness. To this end a more complex perspective than the traditional one (focusing on efficiency, effectiveness and user satisfaction) should be investigated. By ensuring that a learning technology, such as Jarpeb, is relevant to the users' interests and tasks and provokes fun and creativity we reinforce learning activities (such as assessment) and motivate learners helping them to achieve their learning goals.

Elaborating on the aforementioned research insights, our corresponding research hypotheses are:

H6: The Jarpeb system significantly affects the perceived *fun* of the course exercises

H7: The Jarpeb system significantly affects the perceived *interest* of the course exercises

The Jarpeb system

The Jarpeb system prints out student-specific randomized programming assignments, and grades the student's implementation. The requirements that led to the system's design were derived from a mixture of pedagogical and practical constraints:

1. The system should be able to handle multiple relatively-small assignments. To keep students concentrated on the course and provide them with rapid feedback and a sense of accomplishment we handed out a different assignment every week, to be completed in the following 10 days.
2. Every exercise specification had to be different for each student. The difference need not be large, but enough to avoid casual copying and requiring students to understand what they were copying from a friend, and what changes they would have to do to make their friend's solution work for their assignment.
3. The system's structure should allow its implementation by a distributed team. Given the amount of effort required to implement the twelve exercises required for the course (six person/months) we wanted to be able to gracefully split the implementation work among a larger team (four programmers and two testers, in our case).
4. The students should be able to use the system offline. Only a tiny minority of our students have broadband Internet connections; many have PCs at home with a dialup connection, others use the University's lab for downloading Jarpeb. Given these constraints, we designed the system so that it can display a student's assignment and grade its implementation without being connected to a server.
5. Exercise grading should be interactive. A long cycle between submitting a coursework and receiving its grade can demoralize a student. Jarpeb can examine the assignment as implemented by the student and print the corresponding grade, together with a check code. Students can then simply submit their grade and code on a Web form. We decided to risk the chance of students continuously modifying their work until they were satisfied with their grade, to allow them to gain from the gratification of the rapid feedback.

Functional description

Jarpeb is a Java archive file. Students download it from the course's Web site and run it, providing as arguments their unique student identifier number, the course's code, the assignment number, and the operation they wish to execute. The two most important operations are the display of a particular assignment, and the grading of a student's implementation. The following

is a typical example of an assignment. We have set the randomized parts that differ between students in italics:

Course Code: ISDI
Exercise Number: 2
Exercise Title: Function arguments and expressions; loop, and decision statements
Student Id: 8020562

Assignment 1

Write a class named *Rockhair* with a public specifier. The class shall contain a static publicly visible method name *getRewend*. The method shall take two arguments of type *int*, named *tama* and *uru*. The method shall return a value of type *int*. If *lucet* is less than *uru* the method shall return the *product* of the two values, otherwise the method shall return the *square root* of the *sum* of the two values, rounded using the method Math.round().

Assignment 2

On the previous class add a method called *getXanthic* which accepts three arguments of type int: *dryas*, *mouselike* and *aesthete* and returns a result of type int. Loop, *adding 14* times *dryas* to *mouselike* and store the result in a variable of type *int*, called *hypnum*. If the *square* of *hypnum* is equal to *aesthete* return *hypnum*, otherwise return *the negative value* of *hypnum*.

The result of grading the above exercise is something like the following:

Assignment 1: Correct
Assignment 2: Correct
Grade: 10
Verification code: 60c44c40f6999174de52e33fd6a39d20

Design

The design of Jarpeb is based around an interface specification to which all implemented exercises must comply, and a couple of classes providing a common services platform. Each different exercise is a separate class, and its instantiation by a student a distinct object. This architecture allowed us to distribute the implementation work. The common services provided by the Jarpeb platform include:

- the provision of randomized words, integers, floating point numbers, and boolean numbers, based on the student's identifier,
- the calculation of the grade, and the cryptographically strong verification code,
- the interface for displaying and grading the assignments,
- helper methods for dynamically loading a student's implementation and grading it, and
- a system for checking for updates over the web, and notifying the students.

Implementation

The most interesting aspect of the Jarpeb implementation is the use of Java's *dynamic loading* and *reflection* capabilities to grade the student's work. Dynamic loading allows us to load and execute the student's code, and verify that the classes and methods written by the student, directly satisfy the specifications. This provides to the student the flexibility to use any implementation technique that satisfies the exercise's specification and promotes initiative and creativity. Because grading happens within the context of the student's execution environment, students can

not damage or trick the system by creating malicious code. Reflection—the ability of code to examine and reason about itself (Smith 1982)— allows us to examine the student's code for non-functional attributes, like the visibility of methods and fields, or the naming of classes, and thereby provide meaningful error messages during the grading process.

The random identifier names appearing in the assignment are derived from a dictionary, that was filtered to omit illegal Java identifiers. The Java naming conventions are followed for assigning identifier capitalization, but other than that the identifiers have no semantic meaning. The students' reaction to these, often meaningless, exercises was also a subject of our investigation. The system's pseudo-random number generator is seeded with a constant derived from the student's identifier. In this way, individual students will always see the same assignment every time they execute Jarpeb.

The verification code accompanying the grade allows students to complete and grade their exercise without being connected to the internet. The verification code associates a grade with a specific exercise and student, and serves to authenticate the grade when the student enters it on the course's corresponding web form. To satisfy these requirements the verification code is calculated as a cryptographic message digest hash of the exercise's grade, the student identifier, the exercise's code, the course's code, and a shared secret. While this grading method is not immune to reverse engineering attacks, we considered the balance of benefits and risks appropriate for our environment.

Experimental setting

The participants in the Jarpeb instructional setup, apart from the Jarpeb system included the course's instructor, the students, and a separate team of exercise authors and testers. The course's instructor (the paper's first author) structured the syllabus and set the topic for exercises corresponding to each instructional unit (lecture).

The actual exercises were implemented by a separate development team. Implementing a new Jarpeb exercise in not a trivial task. Our exercise implementations averaged 574 lines of commented Java code per exercise; the minimum being 206 lines (a mostly boilerplate "hello world" example), to 1000 lines. In total our exercise implementation code consisted of 6321 commented Java code lines. The structure and development process of Jarpeb allowed us to assign the exercise implementation task to a group of four exercise authors. Each author would get assigned an exercise topic and draft a sample exercise assignment. This would clearly indicate the parts of the assignment that were fixed, and those that were variable—randomized between different students. Authors would then agree with the instructor on the final wording of the assignment, and then write the assignment implementation code. This involves code to print the randomized assignment instructions, and code to load the student's code and grade it, according to the specifications, by using reflective techniques.

We used the concurrent versions system (CVS) to collaborate and to coordinate the work between the exercise authors, testers, and students. Authors could always use CVS to retrieve an up-to-date version of the Jarpeb system's source code. After adding an exercise, or fixing an error, they would commit their change and a compiled version of the program to the system's master CVS repository. Testers and students could retrieve from the repository the compiled version for testing and for actually performing the assignments. The CVS tags each separate exercise with a unique version number. Students and testers could automatically verify that the version of the system they were running was the most recent available, and could otherwise retrieve an updated version of the compiled Jarpeb system.

We assigned a separate group of three testers (laboratory support personnel) to verify the exercises. We found that testing the exercises was a difficult task, mostly because of the

randomization process used to derive separate assignments for each student. For this reason, we assigned to each one of the testers a unique student identifier code—the variable Jarpeb uses to randomize each assignment specification. The testers would download the exercise, and in cooperation with the exercise's author, resolve issues having to do with the exercise's understandability and the correctness of the assignment wording and the grading process. As proof that the exercise was actually completed, testers were required to submit their exercise grade to the instructor. Finally, we used a group of four exceptional students as beta testers. We encouraged those keen Java coders to work on the course's exercises before each exercise was officially assigned and released. We also put the student testers into direct contact with exercise authors to resolve any remaining bugs and problems the students would find. All other students had no way of knowing the exercise authors, and would only get support from the course's tutors.

The course where Jarpeb was employed aims to teach object-oriented programming and the use of domain-specific application programming interfaces (APIs). Its main elements are: programming with objects, constructing classes, inheritance, exceptions, assertion-based programming, interfaces, threads, packages, data structures and their Java APIs, handling XML documents, using regular expressions, file handling, graphic applications, network programming, and database connectivity. The course was taken by second and third year students of the Department of Management Science and Technology at the Athens University of Economics and Business. On the first lecture we told the students that the exercise grades would amount for the majority of their grade (80–90%), provided they would get a passing grade in the final exams. One exercise assignment was given each week; 311 students completed at least one exercise. The course's final exam was taken by 291 students. From the 283 students that got a passing grade in the Jarpeb exercises only 28 failed the exams.

Students taking the exam were asked to complete an on-line questionnaire regarding the Jarpeb exercises (see Appendix A). As an incentive to complete the questionnaire, students who completed it would receive an email message with their final grade, a few days before the grade was officially announced.

It should be clarified that in order to test some of the research hypotheses, students compared the Jarpeb system with traditional systems they used in previous courses. To that end, we had the case of a within-groups experimental setting. The *learning effects* (limitation of a within-groups design) do not affect the reliability of our results since: (a) the manipulated variable is a completely different environment (ie conventional vs. Jarpeb) and (b) the objective of our research is to compare the two environments using students that have used both of them in order to test some of the research hypotheses.

We collected 318 responses; of those 53 were duplicates, typically entered by students who had mistyped their email address. After cleaning another couple of incomplete responses we ended up with 265 valid questionnaires.

Results and discussion

Our sample consisted of 45% male students and 55% female. The 93.1% of the sample (245 respondents) perceived that the Jarpeb system as fair, while 85.1% of them would like the system's use in the introductory course. Finally, 82.5% would like the Jarpeb system to be also used in other courses.

Table 1: t-Test Results for testing the Hypotheses (n=263)

Hypothesis	Questions	Mean (1-5)	*t*	Sig.	*p* Value	Results
H1 *Plagiarism*	#1 and #2	#1= 2.3346 #2= 1.9696	4.505	.000	$p<.001$	Reject H_0
H2 *Understanding of the exercise code*	#3	1.6540	-18.668	.000	$p<.001$	Reject H_0
H3 *Learning the subject matter*	#4 and #5	#4= 3.1103 #5= 3.7072	-7.626	.000	$p<.001$	Reject H_0
H4 *Increases the perceived fairness*	#6 and #7	#6= 3.0152 #7= 2.8479	3.106	.002	$p<.005$	Reject H_0
H5 *Affects the perceived difficulty of the exercises*	#8	3.0837	16.506	.000	$p<.001$	Reject H_0
H6 *Affects the perceived fun of the course exercises*	#9	3.3954	19.528	.000	$p<.001$	Reject H_0
H7 *Affects the perceived interest of the course exercises*	#10	3.4221	18.236	.000	$p<.001$	Reject H_0

As we can see in Table 1 all the null hypotheses (ie 1 to 7) are rejected. This implies that the Jarpeb system significantly:

- *contributes to the reduction of plagiarism (H1)*: students reported that they used their fellow-students exercises in their own exercises significantly less in the Jarpeb system (average 1.96) than in the traditional system (average 2.33) with *p* value <.001.
- *increases the understanding of the exercise code (H2)*: the findings imply that the Jarpeb system helped students to understand exercise code. This finding was provided through a one-tailed *t* test where the students' score (1.65) was compared with the scale's average score . 2.5. The results, therefore, indicate that the null hypothesis should be rejected with a *p* value < .001.
- *contributes to learning the subject matter (H3)*: despite the fact that the average scores of the two settings are quite close, the *t* value (-7.626) and the significance (.000) indicate that the Jarpeb's exercises helped the students to learn the Java language significantly more efficiently in the Jarpeb setting (*p* value < .005).
- *increases the perceived fairness (H4):* to test this hypotheses, we used a modified 1–5 Likert scale. Specifically, first we compared the average scores of the two settings provided by questions 6 and 7 (see Appendix A) in order to test whether there were significant differences between them. As a second step (in case significant differences were found) we checked

which of the two scores was closer to the average score 2.5, which implied that the system is fair. We found significant differences between the two settings and also that the Jarpeb system is perceived as more fair than traditional exercises (average 2.84 with *p* value <.005).

- *affects the perceived difficulty of the exercises (H5):* the findings indicate that the Jarpeb's exercises are perceived significantly more difficult (average 3.08) than the average difficulty score of the scale (question 8). This implies that the Jarpeb system significantly increases the students' perceived difficulty (*p* value < .001) of the exercises in relationship to the material taught within the semester.
- *affects the perceived fun of the course exercises (H6):* as in hypothesis *H5*, the Jarpeb system was found to significantly increase the students' perceived fun (*p* value < .001) since the provided average score (3.39) is higher than the scale's predefined average score of 2.5.
- *affects the perceived interest of the course exercises (H7):* finally, the Jarpeb system was also found to significantly increase the student's perceived interest when working on the exercises (average 3.42 with a *p* value < .001).

In any empirical research design, to have all the null hypotheses rejected is a challenge. In our case the results imply that students perceive the Jarpeb system as an improvement in terms of the investigated dimensions when compared either with their previous experience regarding other assessment methods or with an average situation. It is very promising to have a system that appears to reduce plagiarism while also increasing understanding and learning. Importantly, the students perceive the system as more fair when compared with the established traditional programming assessment systems, and also as fun and interesting. However, a noteworthy disadvantage of the Jarpeb system appears to be that it increases the apparent difficulty of the exercises.

Implications and future research

The present study shows that an e-learning system that creates randomized assignments and automatically grades the corresponding programs can significantly improve the quality of the services provided to the students. Computer science and information technology educators can employ such systems in order to enhance their students learning experience, confront plagiarism, and reorganise their workload. Technology capabilities and the increasing ratio of students to instructors in many universities throughout the world support this course of action. Along these lines, the competition between Universities in terms of building their image and establishing a strong brand name by enhancing their provided services, calls for adopting innovative learning tools, often based on information and communication technologies.

A crucial part when adopting e-learning systems of the type we propose is the testing before their deployment. This test should take place in the specific context that they will be used. For example, an e-learning system used in a Marketing course in a Japanese University may not affect the important learners' dependent variables (e.g. understanding, entertainment, interest, etc.) in the same way that the same system may affect the corresponding dependent variables in an IT course in a University in Brazil. Similarly, the same e-learning system may not be effective for different courses in the same University and, therefore, should not be adopted despite the perceived benefits. Apparently, this is a basic rule that should be followed in any buying decision regarding products or services that are offered to users, business customers, consumers or learners. To that end, decision makers in organizations providing learning services must be very careful in their procurement decisions. Along these lines they should not force the faculty to use e-learning systems only for the sake of improving the image of their organization.

The main limitation of the present study is that the research context of the experiment was an information technology course in a specific University. This implies that the findings may not be

applied either in other courses in the same University or in other educational organizations. To that end, future research should aim at replicating the methodology discussed here in other disciplines and other research contexts, and compare the related findings with the findings provided herein. Furthermore, future research should elaborate more on the *perceived difficulty* dimension in order to design and offer to the students a smoother e-learning experience. Specifically, such study could proceed on running focus groups with experienced e-learning students in order to capture their requirements in terms of the investigated dimension (ie *difficulty* related attributes). Then, the researcher can design alternative e-learning systems and test them against students within laboratory or field experiments. To that end the researcher can employ as the manipulated variable of the causal research design all these attributes that are directly related to the *difficulty* dimension provided through the focus groups and the related literature findings. Finally, further research should also employ other important learners' dependent variables different from those used in the present study.

Acknowledgements

The authors thank Erast Athanasiadis, Stavros Grigorakakis, Vassilios Karakoidas, Polina Tzavela, Victoria Skoularidou, Sofoklis Stouraitis, Vasso Tagkalaki, Vasilis Vlachos, and Giorgos Zouganelis for the work they put into implementing and testing the Jarpeb-based assignments.

References

Aronson, D. T. & Briggs, L. J. (1983). Contributions of Gagne and Briggs to a Prescriptive Model of Instruction. In *Instructional-Design Theories and Models: An overview of Their Current Status*, C.M. Reigeluth (ed.), Hillsdale, NJ: Lawrence Erlbaum Associates, 75–100.

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics & Science Teaching* 20(1), 45–73.

Bloom, M. V., & Hanych, D. A. (2002). Skeptics and true believers hash it out. *Community College Week* 4, 14.

Buck, R. (2002). Typology of Emotions [WWW document]. URL http://wattlab.coms.uconn.edu/ftp/users/rbuck/Uconn9-00/sld001.htm

Cant, S. N., Jeffery, D. R. & Henderson-Sellers, B. (1995). A Conceptual Model of Cognitive Complexity of Elements of the Programming Process. *Information and Software Technology* 37(7), 351–362.

Cernuda del Rio, A. (2004). How Not to Go About a Programming Assignment. *SIGCSE Bulletin inroads* 36 (2), 97–100.

Driscoll, M. (2002). *Web-Based Training: Creating e-Learning Experiences* (2nd ed.). San Francisco, CA: Jossey-Bass/Pfeiffer

Duffield, K. E., & Spencer, J. A. (2002). A survey of medical students' views about the purposes and fairness of assessment. *Medical Education* 36, 879–886.

Ferro, T. (1995). The influence of affective processing in education and training. *New Directions for Adult and Continuing Education* 59, 25–33.

Giuseppe, L. & Brass, D. (2003). Exploring the Social Ledger: Negative Relationships and Negative Asymmetry in Social Networks in Organizations. Forthcoming in *Academy of Management Review*, Special issue: Building Effective Networks, Academy of Management, Chicago, IL.

Goleman, D. (1995). *Emotional intelligence: Why it can matter more than IQ*. New York: Bantam Books.

Hassenzahl, M. (2001). The Effect of Perceived Hedonic Quality on Product Appealingness. *International Journal of Human–Computer Interaction* 13(4), 481–499.

Hazzan, O., & Lapidot, T. (2004). Construction of a Professional Perception in the "Methods of Teaching Computer Science" Course. *SIGCSE Bulletin inroads* 36 (2), 57–61.

Howard, R. M. (1995). Plagiarisms, authorships, and the academic death penalty. *College English* 57, 788–806.

His, S., & Soloway, E. (1998). Learner-centered design. In *Proceedings of the conference CHI 98: human factors in computing systems*, p. 211–212, April 18–23, Los Angeles, California, United States

Hudlicka, E. (2003). To feel or not to feel: the role of affect in human-computer interaction. *International Journal of Human-Computer Studies* 59, 1–32.

Igbaria, M., Schiffman, S. J., & Wieckowski, T. J. (1994). The respective roles of perceived usefulness and perceived fun in the acceptance of microcomputer technology. *Behaviour & Information Technology* 13, 349–361.

Konradt, U., & Sulz, K. (2001). The Experience of Flow in Interacting With a Hypermedia Learning Environment. *Journal of Educational Multimedia and Hypermedia* 10 (1), 69–84.

Love, P. G. & Simmons, J. M. (1997). The meaning and mediated nature of cheating and plagiarism among graduate students in a college of education. *Educational Resources Information Centre* (ED 415 826).

Mayes, J. T., & Fowler, C. J. (1999). Learning Technology and Usability: A Framework for Understanding Courseware. *Interacting with Computers* 11 (5), 485–498.

McConnel, J. J. (1996). Active learning and its use in Computer Science. *SIGCSE Bulletin inroads* 28, 52–54.

Mundorf, N., Westin, S., & Dholakia, N. (1993). Effects of hedonic components and user's gender on the acceptance of screen-based information services. *Behaviour & Information Technology 12*, 293–303.

Newman, I., Daniels, M., & Faulkner, X. (2003). Open ended group projects a 'Tool' for more effective teaching. In *Proceedings of the Australasian Computing Education Conference (ACE2003)*, Adelaide, Australia.

Norman, D. (2004). *Emotional Design*. New York: Basic Books.

Okan, Z. (2003). Edutainment: is learning at risk? *British Journal of Educational Technology* 34 (3), 255–264.

Olson, L. G., Coughlan, J., Rolfe, I., & Hensley, M. J. (2000). The effect of a Structured Question Grid on the validity and perceived fairness of a medical long case assessment. *Medical Education* 34, 46–52.

O'Regan, K. (2003). Emotion and E-Learning. *Journal of Asynchronous Learning Networks* 7 (3), 78–92.

Partala, T. & Surakka, V. (2003). The effects of affective interventions in human–computer interaction. *Interacting with Computers* 16 (2004), 295–309.

Perkins, D & Unger, C. (1999). Teaching and Learning for Understanding. In Reigeluth, C. (Ed.), *Instructional Design Theories and Models Volume II: A New Paradigm of Instructional Theory*. New Jersey: Lawrence Erlbaum Associates, 93–114.

Picard, R., Kort, B., & Reily, R. (2001). *Exploring the Role of Emotion in Propelling the SMET Learning Process* [WWW document]. URL http://affect.media.mit.edu/AC_research/lc/nsf1.PDF

Rea, D. (1998). Achievement motivation as a dynamical system: dancing on the "edge of chaos" with "serious fun." *Educational Resources Information Centre* (ED 415 287).

Reeves, T. C. & Aggen, W. D. (2002). Enhancing E-Learning Assessment and Evaluation Strategies. In *Proceedings of ELEARN 2002* Vol. 2002 (1), 806–811.

Rovai, A. P. (2000). Online and traditional assessments: what is the difference? *Internet and Higher Education* 3 (2000), 141–151.

Schank, R. C. (2001). *Designing world-class e-learning*. New York: McGraw-Hill.

Schneider, A. (1999). Why professors don't do more to stop students who cheat. *The Chronicle of Higher Education* 46, A8–A10.

Smith, B. C. (1982). *Procedural Reflection in Programming Languages*. PhD thesis, Massachusetts Institute of Technology.

Spitzer, D. R. (1996). Motivation: The neglected factor in instructional design. *Educational technology* 36 (3), 45–49.

Worthman, C. (1999). Emotions: You can feel the difference. In: Hinton, A. *Biocultural Approaches to the Emotions*, Cambridge University Press, Cambridge, 41–74.

Appendix A: Questionnaire

1. Please indicate the degree of using your fellow-students exercises in your first year's Java course exercises, in an *1–5* scale *(1: I did not use my fellow-students exercises at all – 5: My exercise was exactly the same with those of my fellow-students)*
2. Please indicate the degree of using your fellow-students exercises in the solution of your Jarpeb's exercises, in an *1–5* scale *(1: I did not use my fellow-students exercises at all – 5: My exercise was exactly the same with those of my fellow-students)*
3. Please indicate whether the direct grading of your answers in the Jarpeb system helped you to understand how the exercise code works, in an *1–5* scale *(i.e. 1: It helped me very much to understand how the exercise code works – 5: It did not help me at all to understand how the exercise code works)*
4. Please indicate whether the first year's Java course typical exercises helped you to learn the Java language, in a *1–5* scale *(1: It did not help me at all to learn the Java language – 5: It helped me very much to learn the Java language).*
5. Please indicate whether the Jarpeb's exercises helped you to learn the Java language, in a *1–5* scale *(1: It did not help me at all to learn the Java language – 5: It helped me very much to learn the Java language).*
6. Please indicate whether you believe that the grading of the first year's Java exercises was fair, in a *1–5* scale *(1: my grade was much higher than the one I deserved, 2: my grade was somehow higher than the one I deserved, 3: my grade was exactly the one I deserved, 4: my grade was somehow lower than the one I deserved, 5: my grade was much lower than the one I deserved).*
7. Please indicate whether you believe that the grading of the Java exercises in the Jarpeb system was fair, in a *1–5* scale *(1: my grade was much higher than the one I deserved, 2: my grade was somehow higher than the one I deserved, 3: my grade was exactly the one I deserved, 4: my grade was somehow lower than the one I deserved, 5: my grade was much lower than the one I deserved).*
8. Please indicate whether you believe that the Jarpeb's exercises were difficult in relationship to the material taught, in a *1–5* scale *(1: very easy – 5: very difficult).*
9. Please indicate whether you perceived the Jarpeb's exercises as entertaining, in a *1–5* scale *(1: no entertaining at all – 5: very entertaining).*
10. Please indicate whether you perceived the Jarpeb's exercises as interesting, in a *1–5* scale *(1: no interesting at all – 5: very interesting).*