Usage of Technology Enhanced Educational Tools for Delivering Programming Courses

http://dx.doi.org/10.3991/ijet.v6i4.1796

M. Ivanović [1], S. Xinogalos[2] and Ž. Komlenov[1]
[1] University of Novi Sad, Novi Sad, Serbia
[2] University of Macedonia, Thessaloniki, Greece

***Abstract*—Methods and approaches behind technology enhanced learning (TEL) in programming courses at a university level encourage continuous research in the last 20 years. Still there is no generally applicable way that would guarantee success. In this paper some experiences gathered during years of a technology-enhanced approach in teaching Object-Oriented Programming (OOP) at two universities in two countries are presented and compared. Emphasis is given on the technology-enhanced educational tools that were selected or developed by the two institutions for teaching OOP. Different traditions and independent development at both institutions allow us to draw generally applicable conclusions and recommendations.**

***Index Terms*—E-learning, Object-oriented programming, Teaching programming, Technology-enhanced learning**

I. INTRODUCTION

Pedagogical theories and methodologies underpinning technology-enhanced learning in introductory programming courses attract teachers' community and foster continuous research at least since the end of the nineties. However, it still has not led to a generally applicable way of teaching and learning that would guarantee the best possible success under the given circumstances. This intrigued and inspired us to analyze, present, and compare selected experiences gathered during nearly a decade of technology-enhanced approach in teaching object-oriented programming languages (OOP) at university level, with specific focus on Java.

The analysis of the approaches and technological tools employed was conducted at two universities in two countries (Serbia and Greece). Our motivation is to try to give recommendations for delivering an introductory course on OOP using TEL. Our institutions apply some of the existing methods and use a set of suitable software tools to enhance everyday teaching practice.

Within CS1 course, at the University of Novi Sad, Faculty of Sciences (UNS-PMF) Modula-2 is used, and at the Technology Management Department at the University of Macedonia (UOM-TMD) the programming language C is used. Imperative first approach is adopted at both institutions. OOP is taught at both institutions within a subsequent course based on Java, a programming language that is widely accepted and proven in practice as a good language and platform for an introductory course on OOP [1]. Thus, students entering the OOP course possess preliminary programming experiences and they are already familiar with concepts and principles of data structures and algorithms.

In this paper we have examined a number of issues affecting learning design and quality of those courses at both institutions. Based on the performed quantitative and qualitative analysis, the history of our two courses and their current state is evaluated in order to extract some key recommendations for planning, designing, deployment, and evaluation of similar courses.

In the following sections we present information regarding the OOP course profile and the application of TEL at UNS-PMF and UOM-TMD. Specifically, in section II we present the aims, the teaching approach utilized and the learning design of the course at both institutions. In section III, we present how both institutions apply TEL at their OOP courses, and in section IV we give some remarks on students' and teachers' experiences and opinions regarding these courses. Finally, some conclusions are drawn based on the experiences and the comparison of the applied approaches at our institutions.

II. COURSE PROFILE

At both **UNS-PMF** and **UOM-TMD** it is required that the majority of second year bachelor students master essentials of Java and OOP and become able to use fundamental concepts of OOP while building at least simple software solutions. Therefore, it was not difficult to identify common goals of our two courses [2]:

- focus on fundamental OO software development tasks and programming concepts rather than simply learning Java constructs;
- comprehending and using standard library classes;
- analyzing/extending existing user-defined classes;
- designing simple OO applications;
- implementing programs in Java.

Students willing to cope with advanced Java concepts, motivated by the requirements from business [3], can further upgrade their Java programming knowledge in subsequent courses offered at their university or by self-studying [1].

A. Teaching Methodology

Teaching methodologies applied within programming courses depend on a wide range of factors: students' motivation, good balance between theoretical and practical aspects of teaching, tradition of teaching at a particular university, teacher's specific teaching style, etc. Even though the university community adopted the object-orientation as an appropriate paradigm with strong expectations, it appears that it is not a particularly easy

task [4] and needs a lot of effort to adjust the teaching style and methodology appropriately.

The traditional method of teaching OOP at **UNS-PMF** was to use the typical face-to-face lectures together with assigning students programming problems to solve. This method, however, worked well only for good students with high analytical problem-solving skills. Even if complemented with an applied, hands-on, approach, it lacks specific mechanisms to provide all the students with equal chances to grasp the programming concepts the instructor aims to convey.

Since UNS-PMF believes that programming skills and techniques should be acquired in interaction with other people and from a wide variety of sources, the current UNS-PMF practice applies blended learning modus by supporting the traditional course with online tools for delivery of self-study instructional units, assignments, topic-specific discussions, various types of online examinations, and other pedagogical aids. Software solutions that are used to support such a delivery of blended OOP course are:

- learning management system (LMS) Moodle [5] for course organization, (adaptive) delivery of additional resources [6], and a variety of communication, collaboration and testing facilities,
- custom-made Web-based tutoring system within the integrated learning environment named MILE [7], which provides additional learning resources and provides high interactivity, offering many examples and exercises,
- IDEs, namely BlueJ [8] and a little bit further in the semester Eclipse [9], for presenting the main concepts of OO design and programming during theoretical exercises, as well as for solving optional homework assignments and students' self-practice in general,
- code visualization tools like Jeliot [10] for providing concrete representation of the dynamic aspects of presented programs and improving students' attention,
- in-house submission system called Svetovid [11], for efficient collection of students' solutions to practical assignments and their timely and efficient grading.

The course at **UOM-TMD** is based on the microworld approach to teaching programming [12] and the educational IDE BlueJ [8]. First, a brief (2 weeks) introduction to OOP concepts takes place based on the microworld objectKarel [13] with the aim of familiarizing students with the most fundamental OOP concepts in a clear and intuitive way. objectKarel is based on Karel++ [14], the well-known metaphor of the world of robots carrying out various tasks in a restricted world. It constitutes a learning environment that incorporates:

- a *learning module* with brief and concise *theory* and *hands on activities* for familiarizing students with the taught concepts before they are asked to implement them,
- a *programming environment*, incorporating a *structure editor* for developing programs, *enhanced error reporting* for the very few syntactical and semantical errors that can arise, *program animation* with immediate feedback on the depicted world of robots and *explanatory visualization*.

Next, the BlueJ IDE and Java are used for presenting the main concepts of OO design and programming. Students use the interactive interface of BlueJ in order to construct objects, invoke their methods and inspect their state without having to write from the very beginning a main method. However, these interactive features of BlueJ are used with caution, since their extensive usage can favor the appearance of specific difficulties and misconceptions regarding the dynamic aspects of OOP [15], [16]. The Jeliot extension for BlueJ is also used for supporting students in comprehending the dynamics of OO programs. In the middle of the course the professional IDE JCreator is also presented and students are left free to decide on their own which environment fits better to their needs.

Overall, the adopted approach is "objects-first" (within the course), iterative (important concepts are taught first and often), and project-driven. The use of the technology-enhanced environments objectKarel and BlueJ plays an important role in applying this teaching approach. Both experience and the results of the long-term evaluation of the course have shown that the combined use of these environments has positive results [17].

As is the case for UNS-PMF, UOM-TMD also uses an LMS, called CoMPUs. However, this is mainly an asynchronous e-learning platform that was implemented for supporting course management for all the departments at the University of Macedonia and does not have any adaptivity features. CoMPUs is used for: organizing and delivering the educational material and additional resources to students, collaboration and communication, assigning projects to students and collecting their assignments.

It can be concluded that both institutions are convinced that programming skills should be best acquired in interaction although some aspects of the actual pedagogical methodology at employed software tools may differ. **UNS-PMF** practices blended-learning modus by offering a wealth of (adaptive) self-study material to students, together with modern and efficient, semi-automated assessment of their work performed in lab using specialized custom tools. **UOM-TMD** focuses explicitly on the proven benefits brought by the microworld approach to teaching programming and the chosen set of tools, together with practicing a project-driven approach within lectures, labs and homework assignments as well.

B. Learning Design

At **UNS-PMF** the course consists of 2 hours per week of lectures, 2 hours per week of theoretical exercises and additional 2 hours per week of lab exercises (during the semester that lasts for 12-13 weeks). In the lectures the teacher explains crucial OO concepts using PowerPoint slides and excerpts of code implementing the concepts presented, while within theoretical exercises students are confronted with complete Java programs/solutions for different problems, as illustrations of the theoretical concepts acquired in the previous lectures.

Within the following lab exercises, attended by 10-15 students at the designated time slot, some practical assignments are solved individually, ranging from very simple and straightforward ones in the first couple of weeks, to rather complex ones at the end of semester.

Their effort is analyzed and graded on weekly basis, so they form one part of their final grades (30%) in small steps. A special custom environment Svetovid [11] is used to provide students with simple mechanisms for editing and testing their code, but also to prevent cheating and allow teachers to grade all the solutions promptly and effectively.

The mentioned final grades are based on max. 30 points for solving practical assignments, and max. 30 points collected in three interim theoretical tests that in fact focus on testing students' problem-solving skills using the newly gained knowledge. Students are required to gather at least 30 out of a total of 60 points to approach the final oral exam (which is worth additional 40 points, i.e. the remaining percentage of the final grade).

At **UOM-TMD** the course consists of 2 hours per week of lectures and 2 hours per week of compulsory lab exercises (during a 13-week long semester). So, in comparison with UNS-PMF the course is taught 2 hours less per week. Actually this means that what is achieved at lectures and theoretical exercises at UNS-PMF must be achieved solely at lectures at UOM-TMD. In the lectures BlueJ's projects and PowerPoint presentations are used. Specifically, each lecture starts with posing a specific real-world problem that has to be modeled with an object-oriented program. A brief discussion takes place in class, in order to identify the classes needed for modeling the system. First, a simplified UML class diagram is presented and then the OO concepts which are used as basis for the implementation of the underlying classes. In several cases students are given excerpts of code implementing the concepts presented, or even brief tests for evaluating their understanding of the presented concepts. In the lab, students solve assignments, with or without the teacher's guidance. Those assignments are sometimes submitted before leaving the lab through the asynchronous e-learning platform CoMPUs. Furthermore, within each lab exercise students are assigned homework that has to be submitted within one week.

Students' final grade consists of: 20 points collected in the lab and by solving homework programming assignments; 20 points from middle-term, open-book exams on paper; 60 points from final-term exams. The aim of the course, as we mentioned in section II, is for students to be able to comprehend and use fundamental OOP concepts for implementing OO programs. So, emphasis is given on lab and homework programming assignments that require several hours of work during the whole semester. The fact that the points granted for these programming projects constitute a small percentage of the final grade is due to the lack of an automated mechanism for preventing cheating. However, experience has shown and students have knowledge of the fact that devoting time to their programming assignments guarantees success in both exams. The middle-term exam has the aim of assessing students' knowledge, while giving both the teacher and the students the chance to acknowledge difficulties and misconceptions and take actions for tackling them in time.

Similar conclusions have been made at **UNS-PMF** as well, but thanks to the specialized tools that are employed to support the efficient, semi-automated assessment of students' progress in online theoretical tests as well as during and after lab exercises during which students produce a lot of code on weekly bases, only 40% of the student's grade is formed in the final oral exam. Homework is given only from time to time, and is not graded, since the focus is on regular evaluation of the work performed in controlled lab environment using custom tools that integrate mechanisms for prevention of cheating. These practical assignments, together with three interim theoretical tests serve, like at **UOM-TMD**, for getting insights in the progress made and acknowledging potential difficulties and misconceptions so that they can be addressed timely and appropriately.

III. TECHNOLOGY-ENHANCED LEARNING

TEL can have a considerable influence on the improvement of students' attitudes towards learning, enhancing their success, increasing communication among students and teachers, and giving them confidence to study advanced subjects without pressure [18]. Recognizing these advantages in teaching programming, both institutions apply blended learning style in their OOP courses, though using different educational tools.

UNS-PMF identified the important goals that should be met when conducting a blended programming course, in particular its online component, in:

- *content organization and presentation, including basic teaching material and additional, flexible self-study resources,*
- *providing efficient communication and evaluation facilities.*

The course conductors at this institution agree with current experiences of other universities that programming languages can be successfully taught in Web-based environments and LMSs. Therefore, LMS Moodle [5] is used for basic course organization and presentation of study material. The course is divided into twenty six sections: sixteen core segments and ten additional ones covering advanced Java topics. It consists of traditional static teaching material, adaptive eLessons, and a mix of various synchronous and asynchronous activities and resources, such as quizzes, glossaries, wikis, and discussion fora. Besides that, some of the resources used during lab exercises are presented, together with practical assignments that are formulated and graded online, but solved individually during regular classes.

For self-studying purposes two possibilities are offered: using eLessons developed in Moodle, extended with basic personalization features [6], and/or using Mag, a custom Web-based tutoring system which is a part of the integrated learning environment MILE [7] that supports teaching, learning and student assessment.

The course can be characterized as learner-centered since, although presenting equivalents to face-to-face lectures, adaptive eLessons implemented in Moodle offer students a possibility to take as much time as they need to explore the available content. They can explicitly choose different paths or can be directed to different parts of the instructional material depending on their answers to the encountered questions, i.e. their previous and newly acquired knowledge.

Mag is as well intended to be used by students who need additional explanations of basic OO and Java concepts presented in a more relaxed and simpler way. As addition to the teaching material Mag offers many simple examples and elements of scaffolding teaching and

visualization that help students in understanding and adopting difficult OO concepts.

For testing students' knowledge two different mechanisms are applied: Moodle's Quiz module for testing theoretical knowledge and problem-solving skills, and special submission system Svetovid [11] for collecting and semi-automatic assessment of students' code, i.e. solutions to programming assignments created during lab exercises. Both solutions help course conductors in preventing various irregularities and obtaining an objective process of grading students' accomplishments.

Within Moodle, small tests for self-evaluation of learning progress are sprinkled throughout eLessons. For grading purposes, three major interim quizzes are conducted during the semester. A pool of over 250 questions has been created, consisting mainly of problem-solving questions similar to those used in Sun's Java Certification Exams. Tests are created and solved using Moodle's Quiz module and in the controlled lab environment, in order to prevent cheating as much as possible. Moodle's gradebook functionality is used for administration of all points and final grades.

To leverage the effort of lab activities and grading students' solutions to programming assignments, teaching assistants use Svetovid. Apart from rather standard functionalities, Svetovid incorporates certain additional characteristics:

- allowing students to code their solutions comfortably with the help of a structure editor with reduced set of functionalities in comparison to other available IDEs;
- allowing beginners to familiarize themselves with the concepts of the specific units rather than writing a program from the beginning (experiments via already prepared tests);
- incorporating significant part of standard Java documentation, extended instruction set and hints, which helps students to quickly refresh things necessary for problem they have been solving;
- detecting and reporting understandable and informative error messages;
- helping instructors to promptly and safely collect and grade student solutions.

Regarding communication and collaboration tools, **UNS-PMF** mainly uses adequate Moodle features (discussion fora, instant messages, integrated mailing facilities, chat sessions, wikis, and blogs). These mechanisms allow students to share ideas, help each other to solve common issues, post their inquiries or reactions to a course (or group) discussion forum, to contact the teachers and get feedback just in time when a piece of advice is needed.

However, as it is proven in literature [19], students are not very eager to use e-learning 2.0 communication/collaboration capabilities like blogs and wikis. Even more discouraging is the fact that less than 20% of students use discussion fora and instant messaging regularly. Most of the students actually report that they still prefer personal communication or use e-mail instead. However, the majority of students agree that employing various ways of communication is indeed very practical and potentially useful.

At **UOM-TMD** teaching and learning is based, as we have already mentioned, on two distinct technology-enhanced programming environments: the microworld objectKarel and the educational IDE BlueJ.

The University's asynchronous e-learning platform called CoMPUs is used for course management and delivery of material. The features of the platform used mostly for supporting students in learning, are:

- *Calendar*: it is kept updated with the lessons carried out, their content, and information for the associated educational material that is available in the platform.
- *Documents*: this tool presents students with a structure of folders corresponding to the lectures. Each folder is enabled when a lecture is carried out and contains all the necessary educational material used at lectures and labs: PowerPoint presentations, programming projects, commented programs and quizzes.
- *Students' assignments*: a tool used for accessing information about the weekly homework assignments, as well as for submitting them in the predefined deadline.
- *Discussion forum*: the forum is organized in sections that correspond to lectures, as well as a section for general topics. Students can post their question/ comment and their colleagues and/or the teacher can respond.

The features of CoMPUs utilized mostly by the teacher for course management are:

- *Description of the course*: this area contains a description of the didactical aims and the content of the course, the available educational material, the textbooks, the software needed, students' obligations and grading policy. This information is available from the beginning of the course and constitutes a clear didactical contract between the students and the teacher.
- *Students' assignments*: the tool is used by the teacher for assigning homework to students, collecting their solutions and downloading them organized in a separate folder per student. A spreadsheet with the assignments submitted by each student can be downloaded, as well as the final grades if the assignments are graded online.
- *Announcements*: announcements can be sent by email to all enrolled students.

Experience has shown that the platform supports both teaching and learning, in various ways: students are always kept informed for all the issues concerning the course (lectures, labs, assignments, exams); the educational material is easily accessible from anywhere; the material can be very easily updated by the teacher depending on students' needs; assignment and submission of projects is simplified; there is a chance for communication and cooperation among students and teachers whenever it is needed. All the aforementioned services are heavily used, except for the last one. Although students are strongly encouraged to use the forum for communication and cooperation during studying and carrying out programming assignments, experience has shown that it is poorly utilized.

In addition, with the aim of supporting students in self-studying, additional educational (SCORM-compliant)

material is being prepared. This material will have the form of a course delivered through the adaptive SCORM compliant LMS ProPer [20]. This material, combined with the advantages of adaptivity, should, at least according to the experiences of **UNS-PMF**, support students that: face difficulties with OOP, do not attend lectures regularly, or need to refresh/elaborate on their knowledge at some time of their studies.

Finally, an issue that has not yet been resolved is the adoption of an existing online tool or the implementation of a new one, for assessing students' programs automatically and providing them with immediate feedback. Such a solution, similar to the one employed at **UNS-PMF**, would certainly bring lots of benefits to both students and teachers conducting lab exercises and evaluating homework assignments.

IV. STUDENT'S AND TEACHER'S EXPERIENCES

During the last several years, at both institutions students regularly fill-in questionnaires to provide feedback for each particular course they take part in, usually towards the end of each semester. Students are usually satisfied with the introductory OOP course, the teaching methodology and the style of grading that teachers employ. They usually point out the great value of such a blended course to anyone who would like to learn basics of OO using Java in a pleasant and interesting way.

The results of the **UOM-TMD** course's formal evaluation that took place towards the end of the winter semester 2010-2011 are presented in Table I. Specifically, the course's average score for seven different criteria, as well as the average score for the 29 winter courses offered at UOM-TMD are presented. For the first four questions the score is in the scale 1-5 (1=not at all, 2=slightly, 3=averagely, 4=much, 5=very much).

The most important conclusions of the course's evaluation can be summarized as follows:

- Students evaluated positively the *organization* and *presentation* (TI.2: 4.26), the *interest* and *contribution* (TI.3: 4.14), the *adequacy of the educational material* (TI.4: 4.22), and the overall *quality* (TI.1: 4.27) of the course.
- The average scores for all the aforementioned criteria were higher for the OOP course than the corresponding average scores for the 29 winter courses, a result that clearly shows students' satisfaction with the OOP course.
- Although students consider the course more difficult than other courses (TI.6), and rather difficult in general, they devote less time to this course each week (TI.7). Taking into account the fact that the number of students that fail the course is not higher than that of other courses considered difficult, as well as the fact that the course is one of the few that employ weekly programming assignments and middle-term exams we believe that the following conclusion can be drawn: *technology-enhanced teaching methodology and educational tools employed at the course are rather successful*.

Similar survey was conducted at **UNS-PMF** as the course's evaluation at the end of the winter semester 2010-2011. The overall grade that the course received was 8.4 on a 1-10 scale. Other specific course's average scores for seven studied criteria are presented in Table II. According to the students' opinions some important conclusions can be made at **UNS-PMF** as well:

- As expected, most of the students felt that they had enough *pre-knowledge* to follow the course (TII.1).
- Students evaluated rather positively the *organization* and *presentation* (TII.2), the *interest* (TII.3) and the up-to-datedness (TII.4) of the course, as well as the *applicability of the knowledge they gained in their future work* (TII.5).
- The *adequacy of the educational material* available online and in paper versions was also ranked high (TII.6).
- Since the survey participants were mainly students that attended the classes regularly (TII.7), we can trust their opinions and their clear overall satisfaction with the course.

TABLE I.
STATISTICS OF COURSE'S EVALUATION AT UOM-TMD

	Question	Average of OOP course	Average of 29 winter courses
TI.1	Was the quality of the course high?	4.27	3.93
TI.2	Was the organization and presentation of the course flawless?	4.26	3.93
TI.3	Was the topic of the course interesting and useful for your studies?	4.14	3.98
TI.4	The educational material (textbooks, notes, exercises, articles etc) was sufficient for the needs of the course?	4.22	3.80
TI.5	Do you attend the lectures regularly? (1=not at all, 2=rarely, 3=quite often, 4 =very often, 5=always)	4.45	4.24
TI.6	Based on your experience from other courses, this course is considered to be: (1=too easy, 2=easy, 3=of average difficulty, 4=difficult, 5=very difficult)	3.80	3.48
TI.7	Besides the lectures and labs, how many hours do you devote to this course each week? (1=<2 hours, 2=2-4 hours, 3=4-6 hours, 4=6-8 hours, 5=> 8 hours)	1.65	1.81

TABLE II.
STATISTICS OF COURSE'S EVALUATION AT UNS-PMF

	Question	Fully agree	Partly agree	Disagree	Cannot tell
TII.1	I had enough pre-knowledge to follow the course.	30%	50%	10%	10%
TII.2	The organization and presentation of the course was flawless.	50%	20%	30%	0%
TII.3	The course's topic and content are interesting.	30%	60%	10%	0%
TII.4	The course's topic and content are up-to-date.	70%	30%	0%	0%
TII.5	The knowledge gained is applicable in practice.	70%	30%	0%	0%
TII.6	The educational material used was appropriate for the needs of the course.	80%	10%	10%	0%
TII.7	I attended the lectures regularly.	60%	10%	30%	0%

- From the answers the students' gave on additional open-ended questions present in the survey it can be concluded that the majority of them consider the OOP course to be more difficult than other courses they attended, mainly because of the amount of work that they are required to perform on weekly basis by solving programming assignments. However, they are aware of the benefits of such practice having in mind their future work that will also depend heavily on practical programming skills and following strict deadlines.
- Knowing that the number of students that fail the OOP course is not higher than that of other courses at **UNS-PMF**, and taking into consideration all the aspects of the course and student's final results, we believe that once again it can be concluded that *TEL and specific tools that were employed within the course proved to be rather successful*.

Teachers that are involved in these courses, claim that, though online activities are fairly challenging supplements to traditional teaching and learning, they can significantly help students. As most of the facilities used for study administration are integrated directly in the chosen LMSs, teachers agree on their positive impact on administrative workload reduction. Additionally, the available mechanisms enable teachers to produce readable, high quality teaching material and improve communication with students. Different communication tools become vital means of informing students on important course issues.

UNS-PMF also has positive experiences with using online tests for official assessment, as well as Svetovid's functionalities, since it prevents students to share their programs during classes, and increases the speed of students' solutions assessment. Students, however, often state that testing conducted using computers causes additional pressure to them, provoked by evident time limits and the possibility of hardware failure. Regarding the use of Svetovid submission system, students sometimes complain that the system is rigid, not comfortable enough in comparison to other IDEs. Therefore, finding a proper balance between students' comfort and needed efficient security measures should be the overall mission for the future course runs.

UOM-TMD also appreciates the support provided both to students and teachers from automated evaluation of students' solutions to programming assignments and is in the process of adopting such a system. However, it does not think that it is necessary to integrate the development environment with the submission and automated evaluation system. It might be more effective to leave the students free to use the IDE that suits better to their needs. Of course, this approach has the disadvantage of potential cheating when the system is used during classes. However, with the evolution of the course this is no more a big issue, since students have realized that cheating does not help them pass the course, while hands-on experience in programming does.

V. CONCLUSIONS

In this paper we presented experiences of teaching OO programming at two universities, comparing the applied approaches and methodologies, assessment strategies and effects that they impose. This comparison allowed us to draw some common conclusions:

- *Objects-first approach is not necessary.* Both directions, top-down (object principles and interfaces presented first) and bottom-up (structured programming and basic building blocks done first) can work well if designed properly.
- *LMSs are necessary for efficient course management and delivery of educational material.* With LMSs the educational material is easily accessible from anywhere and can be instantly updated by the teacher depending on students' needs; students are constantly kept informed for all the issues concerning the course; assignment and submission of projects is simplified; communication and cooperation among students and teachers is always possible. In the case that the LMS utilized is enhanced with adaptivity features the benefits for learning are even greater.
- *Online component is not enough.* Although they can help and motivate students, sometimes online activities are simply not enough. Students are especially reluctant to use contemporary Web 2.0 tools. They still prefer e-mails or face-to-face encounters with teachers when they need additional explanations on some course issues or content.
- *Technology-enhanced educational programming environments support students' introduction to OOP.* **UOM-TMD** gained very positive experience using educational programming environments. The microworld objectKarel supports students greatly in

changing their imperative way of devising solutions to problems to an OO one. Furthermore, BlueJ with its interactive and visualization features supports students in smoothly transferring from the microworld to Java. However, several students, mainly the more motivated ones want to use a professional IDE at some time of the course. Similar conclusions were drawn at **UNS-PMF** as well.

- *Semi-automated evaluation of the work performed by students has positive impact on the effectiveness and fairness of grading process*. **UNS-PMF** gained very positive experiences using online tests and other semi-automatic grading mechanisms, especially in terms of saving teachers' time and increasing the transparency of the evaluation procedure. Although LMSs and other support systems are simplifying the grading processes, it is always advisable to keep the grading schema simple.
- *Homework, promoting continuous work, should count towards grading*. Despite the risk of cheating, **UOM-TMD** finds it good if there are homework tasks significantly contributing to the overall grading. It actually motivates students to work continuously out of the class, promotes online discussions and helps in developing teamwork skills.
- *Continuous monitoring of students' progress is essential for dealing with difficulties timely and effectively*. Monitoring students' progress can be achieved in various ways, such as grading lab activities or homework, middle-term exams, tests and quizzes, etc. Even more beneficial is when these activities are conducted using online tools, together with providing instant, informative feedback.

TEL is *routinely applied* at both institutions, and feedback is positive from both teachers and students. However, it is clear that an institution can successfully employ a *wide variety* of pedagogical methodologies and tools for TEL. There still seems to be room for in-house solutions at large institutions, but open source alternatives are clearly gaining on importance, especially when extended to support adaptability and personalization [21].

Getting and reflecting on valuable *feedback* represents another important issue not ideally resolved yet by large-scale learning environments. However, probably the key research task of today is how and how far should one go in applications of Web 2.0 tools in programming and other university courses right now as well as in the future.

ACKNOWLEDGMENT

This work was produced within the multilateral project "Technology Enhanced Learning and Software Engineering" in which University of Novi Sad (Faculty of Sciences), Masaryk University in Brno (Faculty of Informatics), and University of Macedonia (Department of Technology Management in Naoussa) participate since 2011.

It was also partly supported by the Ministry of Education and Science of the Republic of Serbia within the project no. OI174023: "Intelligent techniques and their integration into wide-spectrum decision support".

REFERENCES

[1] M. Ivanović and T. Pitner, Technology-enhanced learning for Java programming – Duo cum faciunt idem, non est idem. *ACM Inroads*, 2(1), 2011, pp. 55-63.

[2] S. Xinogalos, Guidelines for Designing and Teaching an Effective Object-Oriented Design and Programming Course, In *Advanced Learning*, Raquel Hijón-Neira (ed.), INTECH, 2009, pp. 397-422.

[3] C. Litecky, A. Aken, B. Prabhakar, and K. Arnett, Skills in the MIS job market, *15th Americas Conference on Information Systems*. Association for Information Systems (AMCIS 2009), 6-9 August 2009, San Francisco, California, paper 255, http://aisel.aisnet.org/amcis2009/255

[4] S. Xinogalos, M. Sartatzemi, V. Dagdilelis, and G. Evangelidis, Teaching OOP with BlueJ: A case study, *6th International Conference on Advanced Learning Technologies (ICALT'06)*, 5-7 July 2006, Kerkrade, The Netherlands, pp. 944-946.

[5] H. W. Rice IV, *Moodle 1.9 e-learning course development – A complete guide to successful learning using Moodle*. Packt Publishing, Birmingham, 2008.

[6] Ž. Komlenov, Z. Budimac, and M. Ivanović, Introducing adaptivity features to a regular learning management system to support creation of advanced eLessons. *Informatics in Education*, 9(1), 2010, pp. 63-80.

[7] M. Ivanović, I. Pribela, B. Vesin, and Z. Budimac, Multifunctional environment for e-learning purposes. *Novi Sad Journal of Mathematics*, 38(2), 2008, pp. 153-170.

[8] M. Kölling, B. Quig, A. Patterson and J. Rosenberg, The BlueJ system and its pedagogy. *Journal of Computer Science Education*, 13(4), 2003, pp. 249-268. http://dx.doi.org/10.1076/csed.13.4.249.17496

[9] C. Zhixiong and D. Mar, Experiences with Eclipse IDE in programming courses. *Journal of Computing Sciences in Colleges,* 21(2), 2005, pp. 104-112.

[10] M. Ben-Ari, The effect of the Jeliot animation system on learning elementary programming, *4th Panhellenic Conference on Didactics of Informatics*, 28-30 March 2008, Patras, Greece, pp. 21-30.

[11] I. Pribela, M. Ivanović, and Z. Budimac, Svetovid – interactive development and submission system with prevention of academic collusion in computer programming. *British Journal of Educational Technology*, 40(6), 2009, pp. 1076-1093. http://dx.doi.org/10.1111/j.1467-8535.2008.00894.x

[12] P. Brusilovsky, E. Calabrese, J. Hvorecky, A. Kouchnirenko, and P. Miller, Mini-languages: a way to learn programming principles. *Education and Information Technologies*, 2, 1997, pp. 65-83. http://dx.doi.org/10.1023/A:1018636507883

[13] S. Xinogalos, M. Satratzemi and V. Dagdilelis, An Introduction to object-oriented programming with a didactic microworld: objectKarel, *Computers & Education*, Volume 47, Issue 2, pp. 148-171, September 2006. http://dx.doi.org/10.1016/j.compedu.2004.09.005

[14] J. Bergin, M. Stehlik, J. Roberts and R. Pattis, *Karel++ - A Gentle Introduction to the Art of Object-Oriented Programming*. 2nd edition, Wiley, New York, 1997.

[15] N. Ragonis and Ben-Ari, On Understanding the Statics and Dynamics of Object-Oriented Programs, *ACM SIGCSE Bulletin*, 37(1), 2005, pp. 226-230. http://dx.doi.org/10.1145/1047124.1047425

[16] S. Xinogalos, M. Satratzemi and V. Dagdilelis, Studying Students' Difficulties in an OOP Course Based on BlueJ, *9th IASTED International Conference on Computers and Advanced Technology in Education*, 2006, Lima, Peru, pp. 82-87.

[17] S. Xinogalos, Object-Oriented Programming – What Do Students Think of Objects and Classes?, *14th IASTED International Conference on Computers and Advanced Technology in Education (CATE 2011)*, 11-13 July 2011, Cambridge, UK, pp. 181-186.

[18] W. Horton, *Designing Web-Based Training – How to teach anyone anything anywhere anytime*. John Wiley & Sons, New York, 2000.

[19] S. Downes, E-learning 2.0, ACM eLearn Magazine, October 2005.

[20] I. Kazanidis and M. Satratzemi, Adaptivity in ProPer: an adaptive SCORM compliant LMS. *Journal of Distance Education*

Technologies, 7(2), 2009, pp. 44-62. http://dx.doi.org/10.4018/jdet.2009040103

[21] D. Verpoorten, C. Glahn, M. Kravcik, S. Ternier, and M. Specht, Personalisation of learning in virtual learning environments, Proceedings of the EC-TEL conference, *Lecture Notes in Computer Science*, Vol. 5794, 2009, pp. 52-66. http://dx.doi.org/10.1007/978-3-642-04636-0_7

AUTHORS

M. Ivanović is a full professor at the Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Novi Sad, Serbia (e-mail: mira@dmi.uns.ac.rs). She is author or co-author of 10 textbooks and of more than 200 research papers on multi-agent systems, e-learning and web-based learning, software engineering education, intelligent techniques (CBR, data and web mining), most of which are published in international journals and international conferences. She is also currently Editor-in-Chief of Computer Science and Information Systems Journal.

S. Xinogalos is a lecturer at the Department of Technology Management, University of Macedonia, Naoussa, Greece (e-mail: stelios@uom.gr). He is author or co-author of 47 research papers on teaching and learning OOP, didactics of programming, educational technology and programming environments. He has also designed and implemented two educational programming environments.

Ž. Komlenov is a research and teaching assistant at the Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Novi Sad, Serbia (e-mail: komlenov@dmi.uns.ac.rs). Her research interests include learning technologies, application of intelligent techniques, software engineering, programming languages and tools. She is author or co-author of 15 research papers.

This article is an extended version of a paper presented at the International Conference ICL 2011, held in September 2011 in Piešťany, Slovakia. Submitted, October 30th, 2011. Received 23 August 2011. Published as resubmitted by the authors 22 November 2011.