6/2 (2008), 245–256
tmcs@math.klte.hu
http://tmcs.math.klte.hu
Teaching
Mathematics and
Computer Science

Teaching Java programming using case studies

Zoltán Juhász, Marián Juhás, Ladislav Samuelis and Csaba Szabó

Abstract. The paper deals with the technical background and the pedagogical issues of a specific implementation for the collection, assessment and archiving of the students' assignments written in Java. The implemented system automatically applies object-oriented metrics on the collected works in order to measure the characteristic features of the assignments. Tutors use these results for the detection of plagiarisms and for the selection of outstanding works. The paper interprets the measured values within a real Java course held in the 3rd term of the Informatics bachelor study programme at the technical university. Students have several case studies devoted to the simulation of the ATM (Automatic Teller Machine) at disposal. We conclude that the access to the analyzed pool of case studies, blended with the Sun Learning Connection license from the Sun Microsystems, Inc., is an effective way of teaching programming in Java.

Key words and phrases: Java teaching, object-oriented metrics, blended learning.

ZDM Subject Classification: D60.

1. Introduction

Teaching software engineering subjects through open source-code case studies is not a novel approach in general. In spite of this fact only recently offered sophisticated software engineering courses are based on this approach. This approach is important because industry demands mostly modifications of the existing programs and programs are less frequently built from scratch. Students, who study in this way, could gain complex skills in cognitive processes such as understanding, modification and reuse of the existing software.

This paper is specifically devoted to the topic of teaching Java programming for students who already passed the basic course on object-oriented programming. One way of teaching Java is through small chunks of code, which serve for explaining Java technologies. The other way, which we followed, is to assign independent projects, which cover a scope of lessons the course provides. This variant offers students wider area for experimentation with the gained knowledge. If we choose this possibility, it is very good to have a system that guides students through learning and then stores students' projects in a database for evaluation and later reuse. That is why we implemented a system in order to support these needs. The elaboration of robust courses based on the incremental analysis, the implementation of adequately selected set of open source-code case studies, and the management of students' deliverables for later reuse, all require special attention and effort.

Below, we demonstrate that a well-chosen case study facilitates the introduction of fundamental concepts in a coherent sequence. The basic idea is to guide students through explanations, models and pieces of code on the way to understand a complex code and application, so that they can apply the acquired knowledge in understanding further codes of similar complexity. The results fall broadly into two categories, they are both technical and pedagogical in nature:

(1) technical results
  - Application that guides students through case studies by stepwise refinement. In other words, the developed application provides students and tutors with basic functionalities of a classical Learning Management System (LMS). These functionalities are e.g. logging and tracking of the student's progress. The “pilot” case study, that introduces students into Java, is devoted to the simulation of the Automatic Teller Machine (ATM).
  - Application for archivation purposes and later measurement and evaluation of selected students' projects. After finishing the course (1 semester long) we collect the projects and measure object-oriented characteristics of the projects. The application automatically measures the submitted projects and evaluates the obtained results.

(2) pedagogical results
  - In the project, we were especially interested in the selection of the outstanding works, which should serve for learning purposes in the following academic years. The evaluated data helped during the grading process of the submitted works and at the detection of plagiarism.

The organization of the rest of the paper is as follows: in Section 2 we briefly outline the motivation and related works; in Section 3 we present the experiment and the obtained results. Section 4 shows the summary graphs. Section 5 is devoted to the analysis of the applied metrics and the final section (Section 6) contains a summary and outlines the direction for future extensions of the application.

2. Motivation and related works

So far the typical projects given to students at our department [4], [5], [6], [8] did not match the realities experienced by industry programmers, who deal with much larger systems that contain many features and the code quality expectations are much higher. In addition, in the industry at least 50% of total life cycle costs is devoted to the maintenance of programs, rather than new developments [9].

The major purpose of every e-learning system is to support students in learning and testing their knowledge at the end of the course individually. Building a robust system, which supports this goal is not a trivial task. The obstacles are as follows:

- the provided case studies have to be *attractive*;
- the problem has to be *simply comprehended* and
- the system has to provide an efficient *feedback* both to students and to tutors.

An important characteristic of a case study for the Java introductory course is that it uses some kind of Integrated Development Environment (IDE). Originally we used BlueJ [10] IDE but we did not restrict advanced students to use more sophisticated IDE's like NetBeans [11] or Eclipse [12]. Students were free to use any IDE since we did not insist on students gaining skills in programming event-driven graphical components. A familiar and attractive case-study that provides motivation for understanding of those concepts is described by Buchta et al. [2].

We assembled a case study devoted to the simulation of an Automated Teller Machine (ATM), a selection based on the assumption that the case study has to be simple enough for students to comprehend the task easily. The reason for selecting the ATM simulation was also the fact that it is almost impossible to get open-source from the industry in spite of the fact that the task of the university is also to provide students with real-life problems [1].

This open-source case study was given to students together with detailed explanations of the fields on the class level. In the next step students were asked to experiment freely with the provided case studies (we provided a case study from earlier academic year too but without detailed explanation). The problem of building ATM is simple enough and also sufficiently flexible in order to experiment reasonably. The course built on this idea was created with the aim to enhance students' skills in Java programming and to automatically evaluate their assignments.

As contest in software production became sharp and low-cost applications became extensively implemented, the importance of the productivity and quality in software development are increasing continuously. Measurement plays a critical role in efficient software development, and provides the scientific basis for software engineering as well. These software quality metrics can also help tutors to choose the most appropriate ones for reuse or exhibition of all submitted assignments. One possible reuse of these (submitted) assignments is using them as foundations for creating new case studies. We have focused on metrics, which characterize specifically object-orientation. Of course, the delivered programs can be measured from other points of view, e.g. focusing on complexity metrics as described in the work of Z. Porkoláb [3]. Software complexity metrics are dealt in higher terms and we think that it is more important to rise the awareness of students in the third term to the basic object-oriented metrics.

The implemented system is a result of several capstone projects of students who worked on it almost continuously for 3 academic years. The aim was to provide a sustainable and robust system which could collect outstanding works of students for supporting learning during the following academic years. In this academic year we divided the registered 104 students into 5 groups in order to handle more easily the face-to-face sessions and consultations. We encouraged every student to experiment with the provided code in order to be acquainted with the code and to foster the implementation of new ideas in new classes and methods. The more original the implementation was, the better mark they received. In short, students worked individually and were also marked individually. We were aware of the deficiencies of this approach and we list some of them in the next paragraph.

From tutors' point of view the project's main advantage is having a very comfortable and quick access to every submitted project. Another advantage is the provision of the measured values for grading purposes and checking plagiarism. The electronic archive provides information about the similarity of the assignments and it also offers the values of several object-oriented features obtained from the measurement of the submitted works.

3. Brief description of the e-learning system

The system is implemented in Java using the Tomcat [15] technology and the MySQL [16] server. As mentioned earlier, the implemented system supports students' learning of object oriented programming in Java. It provides a case study, which is divided into 13 lessons. The number of lessons is in accordance with the number of weeks in the semester. Lessons are weaved with small quizzes, in this way students can verify their knowledge interactively. After processing all lessons, students can practice the previously obtained knowledge by creating their own application. These applications are uploaded into the system at the end of the course. The system offers tools for the assessment of the submitted assignments. From the users' point of view, it provides GUI (Graphical User Interface) for 2 groups of users, namely for students and for tutors as well.

3.1. Student's interface of the LMS

At the beginning of the course students have to fill in a registration form for administrative purposes. After submitting this form they have to wait for the confirmation of the registration provided by the tutor. If a student is successfully registered, s/he can login into the system. After the login the actual announcements are at disposal, which are added and updated by the tutor. The lessons are available after the registration.

These lessons support students with learning materials for object-oriented programming in Java step by step. The theoretical background of the case study is complemented by practical examples, so students can obtain a complex knowledge of the problem. After successfully completing all lessons, students should be able to create their own ATM simulation programs, which was described in the practical part of the case studies.

We have automated the submission process in order to archive students' assignments and to evaluate the object-oriented features of their submitted projects.

Students submit the assignments as a JAR file with specific structure in order to facilitate the automatic assessment of their projects. These specific instructions are available for students in detail throughout the course.

3.2. Tutor's interface of the LMS

As we mentioned before, the tutor takes care of registering the students into the system, updating it and creating announcements. The most important issue for a tutor is to have an overview about the submitted projects in order to evaluate them, examine their similarities to avoid plagiarism and select the outstanding works. LMS offers tools for grouping similar projects. This is one way of checking the originality of the works. Every submitted work is evaluated by measuring several object-oriented features. The tutor then selects outstanding projects in order to store them in the pool of outstanding works for the reuse during the next academic year.

4. Experiences with the e-learning system

Registered students already passed successfully a course on programming in C language and learnt the basics of object-oriented programming. For the first time we tested the system with 104 registered students in academic year 2006/2007 in the summer semester of the 3rd term. During that period students had to understand and implement the project using Java. Full-time students created five study groups for consultational purposes and there was one study group with external students. Figure 1 shows the number of students and successful submissions.

As the system operated for the first time this year, the possibility that there would be problems with submitting assignments had been very high. That's why the tutor and senior students assisted continuously during the submission process. Figure 2 shows the success of this submission process.

As mentioned before, the case study contains a tutorial demonstrating the way of creating ATM simulation programs. So the next graph in Figure 3 shows the number of students with assignments similar to the program described in the tutorial of the case study. The higher is the similarity, the higher is the probability of the plagiarism. We note that the tutor had these data at disposal during the grading process.

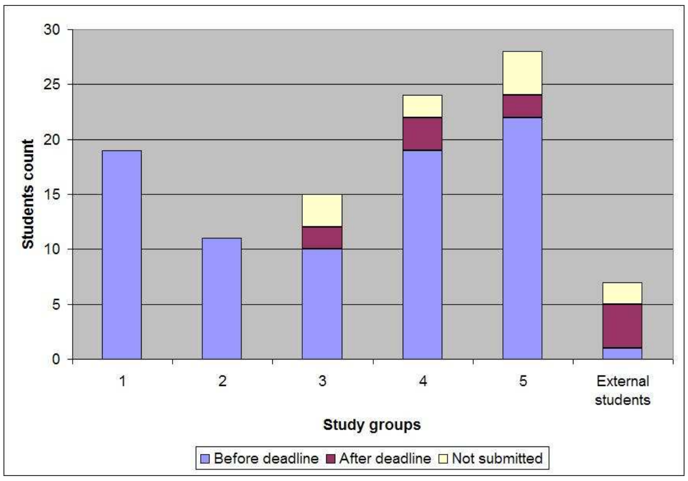

Figure 1. Summary graph of the submitted projects

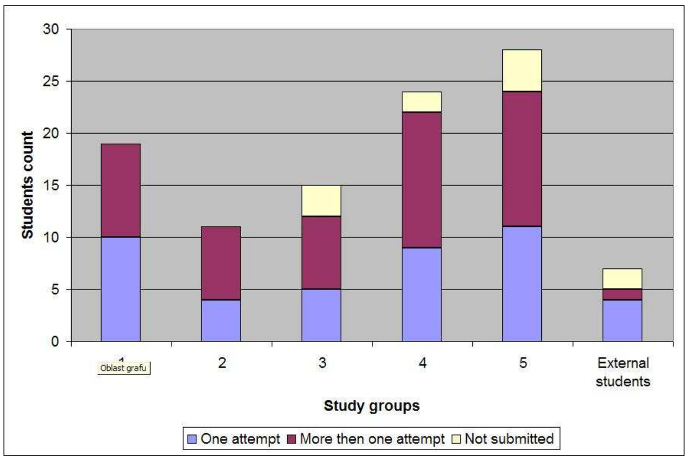

Figure 2. The success of the submission processes

5. Object-oriented metrics applied for measuring the assignments

After uploading the assignments into the system the structure of the JAR file is checked and the student’s program is executed. After this step the assignments is checked and evaluated according to the pre-defined metrics.

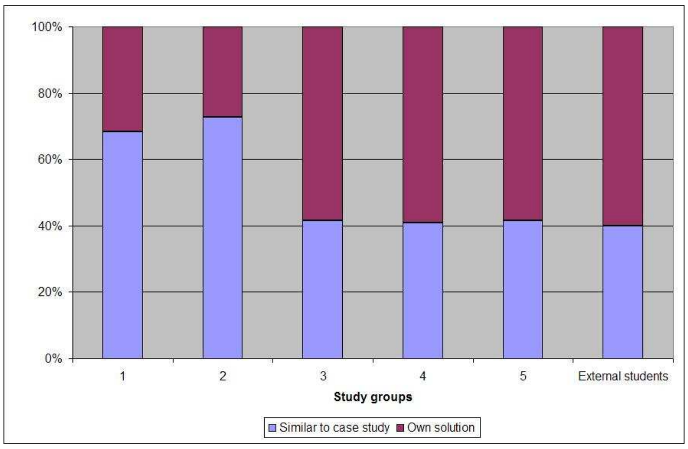

Figure 3. The similarity of the submitted projects to the case study

We used the open source package JDepend [13] for this purpose. It negotiates Java class file directories and generates design quality metrics for each Java package. This approach is automatic and provides interesting data for tutors. The tutor does the final assessment in all cases based on the JDepend package, s/he has the following information about the quality of the source code at disposal [7]:

Number of classes and interfaces: The number of concrete and abstract classes (and interfaces) in the package is an indicator of the extensibility of the package.

Afferent couplings (Ca): The number of other packages that depend upon classes within the package is an indicator of the responsibility of the package.

Efferent couplings (Ce): The number of other packages that the classes in the package depend upon is an indicator of the independence of the package.

Abstractness (A): The ratio of the number of abstract classes (and interfaces) in the analyzed package to the total number of classes in the analyzed package. The range for this metric is 0 to 1, with A = 0 indicating a completely concrete package and A = 1 indicating a completely abstract package.

Instability (I): The ratio of efferent coupling (Ce) to total coupling (Ce + Ca) such as I = Ce / (Ce + Ca). This metric is an indicator of the package's resilience to change. The range for this metric is 0 to 1, with I = 0 indicating a completely stable package and I = 1 indicating a completely instable package.

Distance from the main sequence (D): The perpendicular distance of a package from the idealized line A + I = 1. This metric is an indicator of the package's balance between abstractness and stability. A package squarely on the main sequence is optimally balanced with respect to its abstractness and stability. Ideal packages are either completely abstract and stable ($x = 0$, $y = 1$) or completely concrete and instable ($x = 1$, $y = 0$). The range for this metric is 0 to 1, with D = 0 indicating a package that is coincident with the main sequence and D = 1 indicating a package that is as far from the main sequence as possible.

Package dependency cycles: Package dependency cycles are reported along with the hierarchical paths of packages participating in package dependency cycles.

These features are analyzed automatically in every assignment. The analysis of object-oriented programs involves much more features so we constrain the analysis only to the above-mentioned points. We observed from the submitted assignments that most of the students created only one package and put all classes into one package. This behavior is typical for beginners. These results show that the tutor has to focus on the explanation of the importance of the packages. Quality metric results are very significant because the tutor can reveal students' concepts, which they had to learn from theory and practice. The results of these measurements provide information on how to improve the case studies and the selection of outstanding projects.

We emphasize that the ultimate goal of the evaluation is the selection of outstanding applications, which may serve for learning purposes for students in the following academic years. The tutor is the final judge of the running applications, evaluates its complexity, reviews and analyzes the documentation and/or checks the similarities between the projects (for avoiding plagiarism).

6. Conclusions

As we mentioned in Section 4 the implemented LMS operated for the first time this academic year. Within the course students had 4 different open-source case studies describing an ATM simulation program at disposal. One of them was analyzed in detail. For the successful completion of the course every student had to create his/her own ATM simulation program. After finishing the course, we had approximately further 8 outstanding projects available for learning in the next academic year.

To sum it up, more than 89% of students submitted their assignments successfully. More than 88% of these students submitted their projects before deadline. The most common problem during the submission process was the incompatibility of the JAR files. JAR files created in NetBeans or Eclipse IDEs were not accepted by the experimental system. This is the main reason why only about 46% of submitted assignments were accepted at the first attempt. This obstacle will have to be resolved in the next version of the application. Concerning the similarity of the submitted works, we may conclude that approximately 50% of students tightly followed the code offered by the case studies.

The course was supported with online access to the materials licensed from the Sun Microsystems, Inc. Students had access both to the pool of open-code case studies and to the Sun's materials. This way students obtained complex knowledge by blended learning.

The following extensions are planned:

- enhancing statistic module of the LMS;
- improving the tutor's user interface;
- enhancing the robustness of the LMS in order to accept variants of JAR files.

We conclude that the availability of the online course, licensed from the Sun Microsystems, Inc. [14], blended with the access to the pool of open-source case studies, is an effective way of teaching programming in Java. Of course, we cannot measure their knowledge of Java language but we hope that the more opportunities we provide, the better programmers students become.

Acknowledgements

The research was supported by the following grants:

- Behavioral categorical models for complex software systems. Scientific grant agency project (VEGA) No. 1/0175/08
- Knowledge-based software architectures and their life-cycles. Scientific grant agency project (VEGA) No. 1/0350/08

References

[1] E. Angster, SDP-City against a Vicious Circle!, *First Monday* **9** (2004), December, `http://firstmonday.org` (as of 21.3.2007).

[2] J. Buchta, M. Petrenko, D. Poshyvanyk and V. Rajlich, Teaching Evolution of Open Source Projects in Software Engineering Courses, in: *Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM2006)*, Philadelphia, PA, 2006, 136–144.

[3] Á. Fóthi, J. Nyéky-Gaizler and Z. Porkoláb, The Structured Complexity of Object-Oriented Programs, *Mathematical and Computer Modelling* **38** (2003), 815–827.

[4] J. Genći, *Some consideration about knowledge assessment*, 8th Intenational Conference Virtual University, Bratislava, 13. – 14. 12. 2007, 237–240.

[5] Z. Havlice, *Sybase Tools in the Curriculum of the University*, Sybase Academic User Conference in Budapest, Hungary, 4–5 December 2007.

[6] P. Horovčák and B. StehlÍková, *Complex evaluation of electronic test by both user categories (teacher and also student)*, in: *SGEM 2008 : Modern Management of Mine Producing, Geology and Environmental Protection: 8th international scientific conference*, Albena, Bulgaria, 16–20 June, 2008, 687–694.

[7] R. S. Martin, *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall, 2002.

[8] H. Telepovská, *SQL Statement Knowledge Assesment*, 6th International Conference on Emerging e-Learning Technologies and Applications, Conference Proceedings, Information and Communications Technologies in Learning, Stara Lesna, Kosice, September 11–13, 2008, 181–184.

[9] H. Van Vliet, *Software Engineering: Principles and Practices*, 2nd Ed., (John Wiley & Sons, eds.), West Sussex, England, 2000.

[10] `http://www.bluej.org` (as of 21.3.2007).

[11] `http://www.netbeans.org` (as of 21.3.2007).

[12] `http://www.eclipse.org` (as of 21.3.2007).

[13] `http://clarkware.com/software/JDepend.html` (as of 21.3.2007).

[14] `https://learningconnection.sun.com` (as of 21.3.2007).

[15] `http://tomcat.apache.org/` (as of 21.3.2007).

[16] `http://www.mysql.com/` (as of 21.3.2007).

ZOLTÁN JUHÁSZ and MARIÁN JUHÁS
GRADUATE STUDENTS AT THE DEPARTMENT OF COMPUTERS AND INFORMATICS
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS
TECHNICAL UNIVERSITY OF KOŠICE
SLOVAKIA

E-mail: `juhasz.zoli@gmail.com`

E-mail: `juhasma@gmail.com`

LADISLAV SAMUELIS and CSABA SZABÓ
DEPARTMENT OF COMPUTERS AND INFORMATICS
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS
TECHNICAL UNIVERSITY OF KOŠICE
SLOVAKIA

E-mail: `ladislav.samuelis@tuke.sk`

E-mail: `csaba.szabo@tuke.sk`

(Received April, 2007)