Proceedings of the 7th International Conference on Applied Informatics
Eger, Hungary, January 28–31, 2007. Vol. 2. pp. 55–62.

Teaching Java programming based on the pool of open source case studies*

Ladislav Samuelis, Csaba Szabó, Zdeněk Havlice

Department of Computers and Informatics
Technical University of Košice
e-mail: {Ladislav.Samuelis,Csaba.Szabo,Zdenek.Havlice}@tuke.sk

Abstract

Teaching software engineering through open source case studies is not a new approach. In spite of this fact only recently are offered courses based on this approach. This involves reuse, understanding, modification and extension of existing software. These skills are highly demanded by the industry. The paper deals with the pedagogical and technical background of a specific implementation for the collection, assessment and archiving of the students's projects. The implemented system (3-layer Java based architecture) collects and automatically applies OO metrics in order to measure the characteristic features of the assignments. Results are used for the detection of the plagiarisms and selection of outstanding works. The paper statistically presents the suitability of this approach in a real setting. We conclude that the availability of the online course, licenced from the Sun Mirosystems, Inc., blended with the access to the portal of open source case studies, is an effective way of learning Java.

Keywords: blended learning, open source for teaching software engineering principles, object-oriented metrics

MSC: 68N30

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 are offered sophisticated software engineering courses based on this approach [2]. 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,

*This work was supported by the grant VEGA No. 1/2176/05.

modification and reuse of the existing software. As much as 45% of resources are devoted to testing and simulation [1].

This paper is specifically devoted to the measurement of the object-oriented assignments. 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 into a database for the 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, require special attention and effort.

We show 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 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. loging 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 archivng purposes and later measurement and evaluation of selected students' projects. After finishing the course (1 semester long) we collected the projects and measured the object-oriented characteristics of the projects. The application automatically measures the submitted projects and evaluates the obtained results.

2. pedagogical results

   - We were especially interested in the selection of the outstanding projects, which should serve for learning purposes in the following academic years. The evaluated data helped during the marking process of the submitted projects and at the detection of the plagiarism. We show some snaps of the student's and the tutor's interface during the comunication with the system and the evaluated data.

The organization of the rest of the paper is as follows: in Section 2 we present the experiment and results we obtained. Section 3 is devoted to the analysis of the applied metrics and the final Section 4 contains summary and outlines the direction for future extensions of the application.

2. Brief description of the e-learning system

The system is iplemented in Java using the Tomcat [7] technology and the SQL server. It provides Java case studies, which are divided into 13 lessons. The number of lessons is in accordance to the number of weeks in the semester. Lessons are weaved with small quizzes and students can verify their knowledge interactively. After completing all lessons, students' efforts culminate in 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.

2.1. 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, which is provided by tutor. If student is successfully registered, s/he can login into the system. After login the actual announcements are at disposal. These announcements are added and updated by tutor. The lessons are available after the registration. Figure 1 (some figures' data are in slovak) shows a piece of the text, which analysis a specific class of the ATM simulation program.

These lessons support students with learning materials for object-oriented programming in Java step by step. The theoretical background of the case study is completed with practical examples, so students could obtain a complex knowledge of the problem. After successful completing all lessons, students should be able to create their own ATM simulation program, which was described in the practical part in the case studies. The ultimate goal of this course is that students have to make their own ATM simulation program.

We have automated the submission process in order to archive them and to evaluate the object-oriented features of their submitted projects. Students submit the assignments as a JAR file with specific structure due to the automatic assessment of the projects. These specific instructions are available for students in detail throughout the course.

2.2. Tutor's interface of the LMS

As we mentioned before, tutor takes care about registering students into the system, updating and creating announcements. The most important issue for a

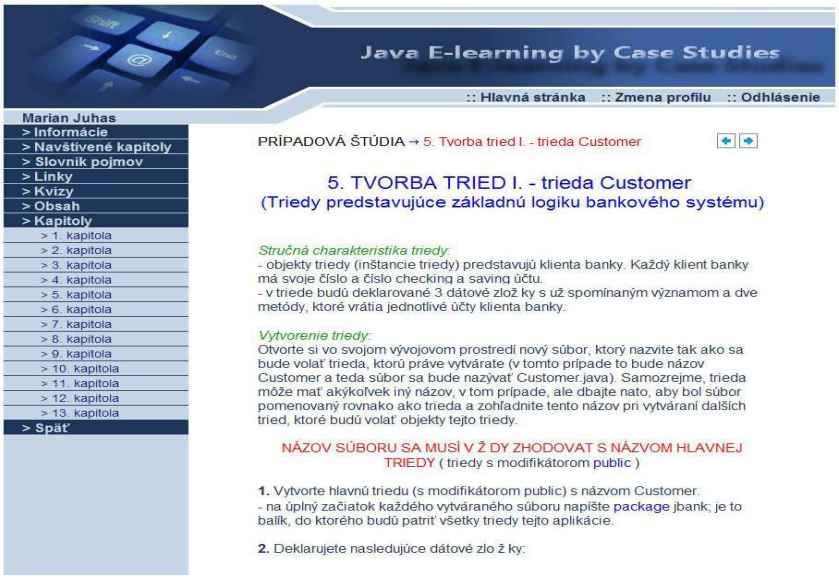

Figure 1: Student's interface of the LMS

tutor is to have an overview about the submitted projects in order to evaluate them for similarities and selection of the outstanding works. LMS offers tools for grouping similar projects. This is one way how the tutor can check the originality of the projects. Every submitted project is evaluated by measuring several object-oriented features. Tutor then selects the outstanding projects in order to put them into the pool of outstanding works for the reuse in the next academic year.

2.3. Experiences with the e-learning system

Registered students already passed successfully programming in C language and basic 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 consultacy purposes and there was one study group attended by external students.

As the system operated for the first time this year, the possibility that there will be problems with submitting assignments was very high. That is why the tutor and students from higher terms assisted continuously during the submission process.

As mentioned before, case study contains a tutorial how to create ATM simulation program.

3. The object-oriented metrics applied for measuring the assignments

The software development process is no doubt a complicated one. The end product follows a chain of analysis, design, development and testing process. At each stage, it is important to follow a well-defined methodology to ensure a quality end product. For large scale projects, each stage in the whole process is a challenge. In this context, the software design and coding metrics play an important role in ensuring the desired quality.

3.1. The applied Object Oriented Metrics

After uploading the assignment into the system it is checked against the structure of the JAR file and then executed on a local PC. After this step the application is checked and evaluated against the predefined metrics. We used the open source package JDepend [5] for this purpose. It traverses Java class file directories and generates design quality metrics for each Java package. This approach is automatic and provides interesting data for tutors. Tutor does the final assessment in any case. On the basis of the JDepend package, tutor has at disposal the following information about the quality of the source code:

3.1.1. Number of Classes and Interfaces

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

3.1.2. Coupling

Coupling is a measure of interdependence of components. High coupling implies strong interconnections between program units, while loose coupling implies independence. Object-oriented designs reflect the real world as independent objects, which leads to loose coupling, if designed well. Once again, inheritance causes tight coupling among classes. Because inheritance and polymorphism are heavily used in object-oriented paradigm, a study of metrics for object-oriented software becomes an important research project [9]. JDepend offers these kind of couplings:

- **Afferent Couplings (Ca)** - The number of other packages that depend upon classes within the package is an indicator of the package's responsibility.
- **Efferent Couplings (Ce)** - The number of other packages that the classes in the package depend upon is an indicator of the package's independence.

3.1.3. Abstractness - (A)

You measure the abstractness of a package by calculating the ratio of the number of abstract classes (and interfaces) in the package to the total number of classes in

the package. Abstractness can be measured using the following formula: A=Na/Nc

- **A** - represents a package's abstractness.
- **Na** - represents the number of abstract classes and interfaces in a package.
- **Nc** - represents the number of concrete classes in a package.

An abstractness value of zero indicates a completely concrete package, whereas a value of one indicates a completely abstract package.

3.2. Why to measure student assignments?

Analysis of object-oriented programs involves much more features and we constrain the analysis only to the above-mentioned points. We observe from the submitted assignments that most of students created just one package and put all classes into one package. This behavior is typical for beginners. These results show that tutor has to focus on the explanation of importance of the packages. Packages are more scalable, flexible, it is easier to read complex code through packages. Quality metric results are very important because tutor can reveal students' concepts, which they had to learn from theory and practice. Results of these measurements provide information how to improve the case studies and the selection of outstanding projects. We stress that the ultimate goal of the evaluation is the selection of outstanding applications, which may serve for the learning purposes for students in the following academic years. Tutor is the final judge of the running applications, evaluates the complexity, reviews and analyzes the documentation or checks the similarities between the projects (plagiarism). At the moment, the metrics we provide is used to make the tutor's evaluative process easier and it is not useful for students at all. Our plan is to offer the advantages of software measurement for all users and it should be another teaching aid for students. This could improve assignments design quality. Pure numbers do not give any sense for students, hence it follows the brief explanation of metrics meaning has to be also provided. Our aim is to extend collection of metrics measured by system namely Chidamber and Kemerer metric (henceforth, CK). Use of CK set of metrics and other complementary measures are gradually growing in industry. The object-oriented metrics proposed by Chidamber and Kemerer can be summarized as follows [8]:

- **Weighted Methods per Class (WMC)** - this is weighted sum of all the methods defined in a class.
- **Coupling Between Object Classes (CBO)** - it is a count of the number of other classes to which a given class is coupled and, hence, denotes the dependency of one class on other classes in the design.
- **Depth of the Inheritance Tree (DIT)** - it is a length of longest path from a given class to the root class in the inheritance hierarchy.

- **Number of Children (NOC)** - this is count of the number of immediate child classes that have inherited from a given class.
- **Response for a Class (RFC)** - this is count of the methods that can be potentially invoked in response to a message received by an object of a particular class.
- **Lack of Cohesion of Methods (LCOM)** - a count of the number of method-pairs whose similarity is zero minus the count of method pairs whose similarity is not zero.

These metrics should offer reinforcement that your software design is robust. While good metrics do not guarantee a quality design, good metrics do help bolster confidence. When used judiciously, these software metrics can be a valuable aid in assessing quality design.

4. Conclusions

To sum up, students had at disposal within the course 4 different open-source case studies describing an ATM simulation program. 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 have got approximately further 8 outstanding projects available for learning in the next academic year.

More than 89% of students successfully submitted their assignments and more than 88% from these students submitted their projects before deadline. The most common problem during the submission process was the incompability of the JAR files. JAR files created in NetBeans [3] or Eclipse IDEs [4] were not accepted by the experimental system. This is the main reason why only about 46% of submitted assignments were accepted on the first attempt. This obstacle has to be resolved in the next version of the application. What concerns the similarity of the submitted works, we may conclude that approximately 50% of students tightly followed the code offered by case studies.

The course was supported with the online access to the materials licensed from the Sun Mirosystems, Inc. [6], Students had access both to the pool of open-code case studies and to the Sun's materials. In this way students obtained blending learning.

The further planned extensions are:

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

We conclude that the availability of the online course, licensed from the Sun Mirosystems, blended with the access to the pool of open-source case studies, is an effective way of teaching programming in Java.

References

[1] Cartwright, M., Shepperd, M., IEEE Transactions on Software Engineering, *IEEE Computer Society Press*, (2000), 786–796.

[2] Buchta, J., Petrenko, M., Poshyvanyk, D., Rajlich, V., Teaching Evolution of Open- Source Projects in Software Engineering Courses, *Proceedings of 22nd IEEE International Conference on Software Maintenance* (ICSM'06).

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

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

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

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

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

[8] Subramanyam, R., Krishnan, M. S., Epirical Analysis of CK Metrics fo Object-Oriented Design Complexity: Implications for Software Defects, *IEEE Transactions on Software Engineering*, vol. 29, No.4, (April 2003).

[9] Liu, X., Object-Oriented Software Metrics, Department of Cornputer Science University of Manitoba Winnipeg, Manitoba, Canada (1999).

Ladislav Samuelis, Csaba Szabó, Zdeněk Havlice
Department of Computers and Informatics
Technical University of Košice
Letná 9
04200 Košice
Slovakia