Studying the process of software development through intensive use of software tools

Violeta Bozhikova[1], Mariana Stoeva[2] , Dimitrichka Nikolaeva[3], Zlatka Mateva[4]

Abstract – **The paper focuses on the further development of our approach to teach the discipline of Software Engineering (SE) which is based on intensive use of software tools. The use of software tools in the process of software development is a way to increase both the effectiveness of teaching SE and the effectiveness of the software engineer's work. This approach was used during the "Software Engineering" course labs at the Technical University – Varna. The approach is intended to focus students' attention on both the use of software tools in all phases of software developmenton as well on the tools development. An open for future development package of software tools is developed for this case; its current structure is discussed in the paper.**

Keywords – **Software Engineering, Software Engineering education, tools for Software Engineering education, tools supporting software development, etc.**

I. INTRODUCTION

"Software engineering" (SE) is a basic course for bachelor degree students in "Computer Science". It examines the different models of software life cycle and covers all phases and activities of the software production. The course is difficult for the students because of its highly theoretical nature. The popular SE books also do not easily attract the students' attention, because of their theoretical-style approach. As a result of everything mentioned above, students lose interest in the Software Engineering discipline.

Based on our long teaching experience in the field of Computer Sciences in the Technical University in Varna, we believe that teamwork and tool supported project-oriented exercises help students easily understand the theory included in SE textbooks.

This paper is about the further development of our approach to teach Software engineering discipline, which is based on intensive use of software tools ([3] and [5]). An open for future development package of software tools ([1] ÷ [5]) is in the core of this approach; its current state is discussed in the paper.

[1]Violeta Bozhikova is with the Department of Computer Sciences and Engineering, Technical University of Varna, 1 Sudentska Str., Varna 9010, Bulgaria, E-mail:vbojikova2000@yahoo.com.

[2]Mariana Stoeva is with the Department of Computer Sciences and Engineering, Technical University of Varna, 1 Sudentska Str., Varna 9010, Bulgaria, E-mail: mariana_stoeva@abv.bg.

3Dimitrichka Nikolaeva is a PhD student in the Department of Computer Sciences and Engineering, Technical University of Varna, 1 Sudentska Str., Varna 9010, Bulgaria, E-mail: dima.nikolaeva@abv.bg

[4]Zlatka Mateva is is with the Department of Computer Sciences and Engineering, Technical University of Varna, 1 Sudentska Str., Varna 9010, Bulgaria, E-mail: ziz@abv.bg

II. OUR APPROACH

In this section we discuss our approach to teach the discipline of Software Engineering which is based on intensive use of self developed software tools.

The discipline of Software Engineering is conducted in the third year of the student's curriculum. It brings 6 credits and includes lectures, practical labs and course project. The students work in teams of maximum 3 students, each student has his different role in the team.

In the first week, all teams receive by the teacher a list with practical software problems to resolve. The last week, each team must submit implementation and documentation of the entire list of software problems.

The student semester includes 15 weeks. The software tools used in the different phases of the software life cycle by weeks of semester are shown below in the Table 1:

TABLE I
THE SOFTWARE TOOLS USED IN THE DIFFERENT PHASES OF THE SOFTWARE LIFE CYCLE BY WEEKS OF SEMESTER

Tool	SE Phase	Weeks
SCE	Requirement Analysis and Specification	Week 1 ÷ Week 2
SR&A	Operation & Maintenance	Week 13 ÷ Week 14
VLT	Design, Coding and Testing	Week 6 ÷ Week 12
PMS	Software Project Management	Week 1 ÷ Week 15
DPAP-T	Design, Coding and Testing	Week 6 ÷ Week 12
UML-T	Requirement Analysis and Specification	Week 3 ÷ Week 5

The teams work is based on evolutionary model for software development. The package of software tools (fig.1) used in solving the software problems is extensible and currently consists of six tools, supporting different software development activities (fig.2):

Fig 1. The package of software tools

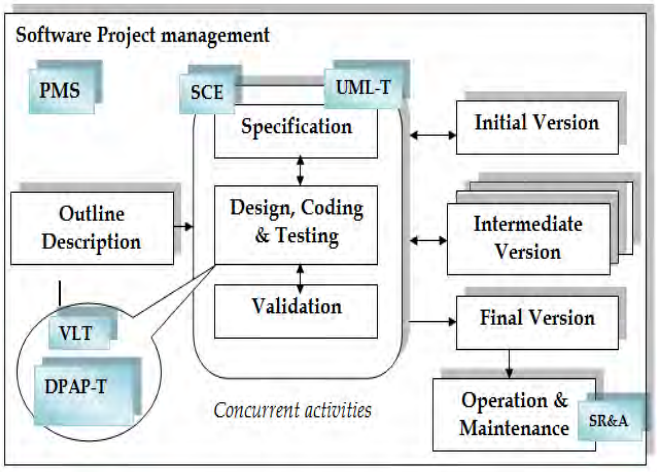

Fig. 2. Software tools in different phases of the software life cycle

- A tool that supports Software Project Management, named PMS.

This tool was in details discussed in ([3] and [5]). Project management software (PMS) is a term covering many types of software, including scheduling, cost control and budget management, resource allocation, collaboration software, communication, quality management and documentation or administration systems, which are used to deal with the complexity of large projects.

- A tool (VLT) with lecture notes, exercises and tests on programming.

VLT ([3] and [5]) is a case of educational software that aims at helping the student study the relevant Microsoft.Net programming language. Before doing the current laboratory exercises, the students could read the language materials. Choosing "Tests" pane the students could make the corresponding quiz and evaluate their learning progress. The teacher also can analyse the students' results, and get an overall view for the progress of the students. The architecture of the developed web-based tool provides instructors a possibility to easily create and manage different course materials. Although the tool doesn't impose limitation about the program language materials, the students are suggested to use a .Net language.

- A tool (SCE) that supports the software cost estimation ([3], [4] and [5]).

This tool implements a hybrid approach for software cost estimation [4], which is based on the classical models for software cost estimation: Basic COCOMO, COCOMOII and Function Points Analysis. Software estimation is the part of project planning aimed at size estimation, effort, time, people required etc. Software cost estimation process gives the information needed to develop a software project's schedule, budget and assignment of personnel and resources.

- A tool for Software Re-structuring and Analysis (SR&A) is integrated in the package ([2], [3] and [5]).

This tool could support mainly the maintenance activity. Using this tool the students could analyze and re-structure the software with the idea to improve its structure. The software must be formally presented as a weighted oriented graph G=(X, U), where the set of weighted nodes X (N=|X|) models the software's components (classes, modules, files, packages, etc.) and U (the set of graph's edges) presents the dependencies between the components (e.g., procedural invocation, variable access, etc.).

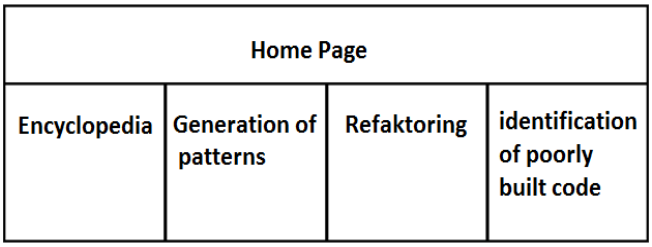

Fig. 3. The structure of DPAP-T

- A tool for software anti-pattern identification, named DPAP-T.

DPAP-T is a web-based application that contains home page and several sections representing different aspects of design patterns and anti-pattern support (Fig. 3).

Home page aims to present the different sections of the system with a short description and redirect the user to any of them.

Encyclopedia: it is a section of the system, which aims to provide information about design patterns and anti-patterns. This section describes the problems that each design pattern resolves; it describes the benefits of the use of design patterns and the situations in which it could be used. Anti-patterns are common solutions to problems that lead to negative consequences. As a result, they produce a poor code structure. The structure of the code is an important factor in maintaining and adding new functionality in one system.

Generation of patterns: this section provides functionality for design pattern generation (fig 4). An example of design pattern (abstract factory) generation is presented in figure 5.

Section "Refactoring" provides methods for automatic code refactoring. Refactoring is a reconstruction of the code with the goal to improve its properties. The code is supplied as input to the refactoring method. The method performs the appropriate changes and returns as a result the modified code. 8 refactoring methods are provided by the tool: "Extract Method", "Inline method", "Replace Temp with Query", "Encapsulate Field", "Replace Magic Number with Symbolic Constant", "Replace Constructor with Factory Method" and "Self Encapsulate Field".

Identification of poorly built code: this section offers methods for analysis of the code. It is supplied as input of a

particular method which identifies the poorly constructed code sections. The poorly build code could be rewritten to be more easily readable and to allow easy maintenance. 3 methods (fig.3) for poorly built code identification are provided by the tool: “Duplicated code”, “Too many parameters in a method”, “Complicated If’s”. Several algorithms that improve the software readability, software maintenance and the ability to add new functionality are provided, but still are waiting to be realized.

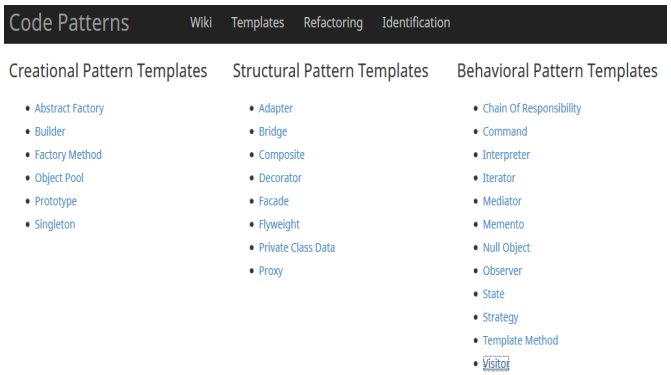

Fig. 4. Section “Generation of patterns”

Fig. 5. An example of design pattern (abstract factory) generation

- A tool for UML diagrams training and drawing, named UML-T.

This software tool (UML-T) provides theoretical and practical training for Unified Modeling Language (UML). It contains 2 modules: “UML textbook” and “UML editor” - (fig.6 and fig.7).

UML is a "universal" to the process of software development language for visual modeling. The visual modeling (visual modeling) is a method that:

- Use graphic (often based on graphs) model for software visualization;
- Offers subject area modeling from different viewpoints (perspectives);
- Supports almost all phases and activities of the software development process; it can be used for both: the development of new software and the evolution of the existing software.

The diagrams are the most important building elements in UML which can be used for the purposes of the:

- Specification of the system requirements and the system architecture;
- System documentation;
- Model simulation (with automatically generated code);
- Model testing and validation;
- Communication between project participants, etc.

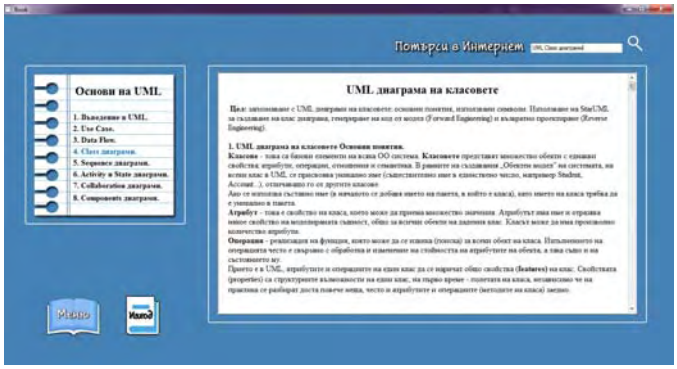

Fig. 6. UML textbook

UML-T supports the drawing of the basic UML diagrams trough “UML editor” module. It provides an intuitive and very simple interface. Compared with the existing commercial and free UML tools, UML-T provides a training subsystem (something as e-book for UML), supports the modeling of Data Flow diagrams, but it is still in its initial phase of development and does not yet support the entire set of charts and Reverse/Forward Engineering options. As a conclusion, many things in the current realization of UML-T wait to be improved.

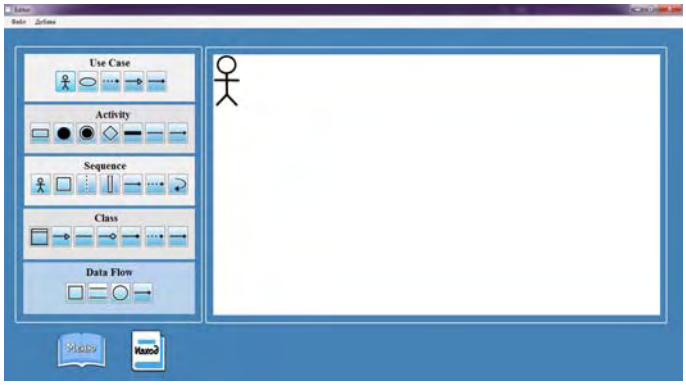

Fig. 7. UML editor

III. CONCLUSION

The paper focuses on the further development of our approach to teach the discipline of Software Engineering which is based on intensive use of software tools. An open for future development package of software tools is in the core of this approach; its further development is discussed in this paper.

Our observation is that the effect of the approach application for SE educational purposes is positive. The student's interest in SE discipline has grown. It is intended to focus students' attention on both the use of software tools in all phases of software developmenton as well on the tools development. The students acquire useful theoretical and practical knowledge in software analysis and specifications, software cost estimation, software maintenance, software development and so on.

Further work is needed in order to improve and to enrich the teaching package: the existing tools could be improved and unified; new functionalities could be aided; new tools could be included in the package.

ACKNOWLEDGEMENT

The authors express their gratitude to the students Bozhidar Georgiev and Nely Evgenieva, who participated actively in the development of the package resources. The paper was sponsored by two research projects in TU Varna: НП8 and ПД6.

REFERENCES

[1] Violeta Bozhikova, Mariana Stoeva, Krasimir Tsonev, A practical approach for software project management, CompSysTech Conference 2009, Russe (to appear)

[2] V.Bozhikova, M. Stoeva, A. Antonov, V. Nikolov, Software Re-structuring (An architecture-Based Tool), ICSOFT 2008, Third Int'l Conference on Software and data Technologies, pp.269-273, Porto, Portugal, 2008.

[3] Bozhikova V., N. Ruskova – "A Computer-Based Approach for Software Engineering Teaching", Proceedings of the International Conference "E-Learning and the Knowledge Society", Berlin, GERMANY, 2009, pp. 161-166, ISBN 1313-9207.

[4] Bozhikova V., M. Stoeva – "An Approach for Software Cost Estimation", Proc. of the International conference on computer systems and technologies (CompSysTech'2010), International Conference Proceedings series - V.471, София, Bulgaria, 2010, pp.119 -124, ACM ISBN 978-1-4503-0243-2.

[5] Bozhikova V., M. Stoeva, N. Ruskova – "A tool package for software engineering teaching", Материалы XI Международная конференция „Стратегия качества в промышлености и образовании", Том 1, Варна, 2015, pp. 313-318.