An Approach to Teaching "Software Design Patterns"

Violeta Bozhikova[1], Mariana Stoeva[2] and Veneta Aleksieva[3]

***Abstract* – This paper is about some problems of teaching "Software Design Patterns" in the Master's degree education and our approach to teaching this discipline. The paper underlines traditional and specific requirements respected by the approach developed and presents its main characteristics, based on the method of electronic textbook. Then, the approach for self-testing and examination of knowledge is discussed and some ideas for further development of the testing part are marked.**

***Keywords* – Electronic Textbook, Computer-based Learning, Teaching Strategy, Knowledge Testing and Evaluation.**

I. Introduction

Design patterns are attracting more attention now because they encapsulate valuable knowledge to solve recurring design problems and improve the quality of programming work. These architectural constructions for reuse become popular after 1994 with the book of Erich Gamma and al. [2]. A design pattern provides a general solution to a common problem in software design and is a language-independent description (or template) of the problem in general, which can be directly transformed into a code.

This paper is about our approach to teaching „Software Design Patterns. The course „Software Design Patterns" is included as an elective, in the second semester of the master's degree program for specialty Computer Systems and Technologies (CST), in the Technical University of Varna. The course includes lectures and practical exercises. The main topics of the course are related to the study of the three main groups of patterns (building, structural and behavioural) and the ways of their multiple use, their combination, their documentation and testing. Unified modelling language (UML) is mainly used for the pattern presentation.

It is seen that the course is based on extensive field of knowledge. Students should know the principles of object-oriented analysis and design, the unified modelling language (UML) and at least one object-oriented language. There are a lot of problems in teaching such a discipline: how to motivate the students to learn this complex technology, how to overcome the problem associated with the different background of the students, how to solve the problem of their employment (the majority of students enrolled in Masters Education are employed). A flexible teaching approach must be developed, an approach that encourages the students, regardless of their different background, an approach that holds the students attention and complies with their employment, providing them and individual workspace.

Section 2 presents the main characteristics of the proposed approach which is based on the method of electronic textbook: traditional and the specific requirements, respected by the teaching approach; structure and implementation of developed electronic textbook and specially – our approach to check the learning.

The last section presents our conclusions and future work.

[1]Violeta Bozhikova is with the Faculty of Computing and Automation, TU-Varna, Studentska 1, 9010 Varna, Bulgaria, E-mail: vbojikova2000@yahoo.com.

[2]Mariana Stoeva is with the Faculty of Computing and Automation, TU-Varna, Studentska 1, 9010 Varna, Bulgaria, E-mail: mariana_stoeva@abv.bg

[3]Veneta Aleksieva is only presenter of this paper and is with the Faculty of Computing and Automation, TU-Varna, Studentska 1.

II. Our approach to teaching „Software Design Patterns"

A. Traditional and specific requirements to electronic teaching materials

We offer an approach for teaching software design patterns based on the electronic textbook. E-books (electronic textbooks) are computer-based systems [3, 4], mainly oriented towards training students, also examination and knowledge evaluation of the students. To electronic teaching materials have both traditional and specific requirements. As "traditional" we could define the following properties:

- Adequacy (in terms of curriculum) and completeness of the statement;
- Logic and coherence of the presentation.
- Accessibility statement of material
- Scientific character of the material.

To achieve the objectives of the course „Software Design Patterns" and to overcome the above problems, the electronic textbook, proposed by us has a structure shown in Figure 1 in order to respect the following specific requirements:

- To focus on the practical aspects of applying patterns in software development instead of theory: the aim is to motivate the students to use this technology and maintaining enough self-discipline to understand its general ideas. During laboratory exercises, students learn patterns through small examples check their knowledge (through tests) and finally - develop specific software tasks with higher complexity. The aim is, after graduation, students are able to independently design, develop, and document complete software solutions using a combination of patterns.
- To be based on examples written in C #.Net, a language that is studied in the previous semester in the Master Course.
- To focus mainly on individual work rather than teamwork.
- To be bilingual (to support Bulgarian and English language) in order to serve as a textbook on the subject "Software Design Patterns" for Bulgarian-speaking and English-speaking students.

- enable self-testing and self-assessment of the student's knowledge at different levels; for each topic, for each module and for the whole material;
- be convenient for maintenance and future evolution; Since software design patterns are numerous, and the developed electronic textbook focuses on the universal patterns only [5], the architecture of the electronic textbook is simplified to allow easy further development without impairing the quality in. Each topic presents a pattern and is stored in separate .rtf file when the tests are stored in .txt files (shown in the left part of Figure 2). Thus the change and complete replacement of the content of a pattern and the adding of new patterns is easy, with minimal technical effort, without program intervention in the system itself.

B. Structure and implementation of developed electronic textbook

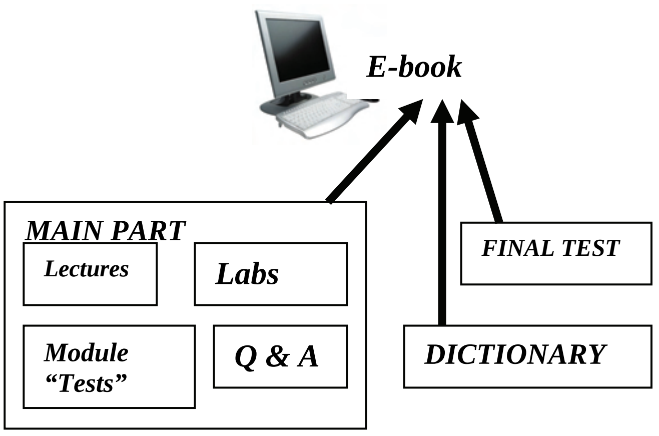


Fig. 1. The generalized structure of the created electronic textbook

The main modules (see figure 1) of the created electronic textbook on the subject " Software Design Patterns" are:

- MAIN MODULE: the most important part of the book, which exposes the contents of the course on modules and themes, each theme includes descriptions of lectures and laboratory exercises. This part is divided into three main modules, according to the three main groups of the patterns (building, structural and behavioral). Each lecture and each laboratory exercise are accompanied by graphic illustrations (UML diagram, etc.). Each laboratory exercise includes a description of assignments of tasks to be performed in order to assimilate the material in practice. This part includes also Q & A MODULE with frequently asked questions and answers.
- TEST MODULE: includes questions about self-assessment at different levels: at topic level, module level and final test level;
- MODULE DICTIONARY: includes a glossary of terms;

Figure 2 shows the main window of the electronic textbook, realised as a desktop C#.Net application.

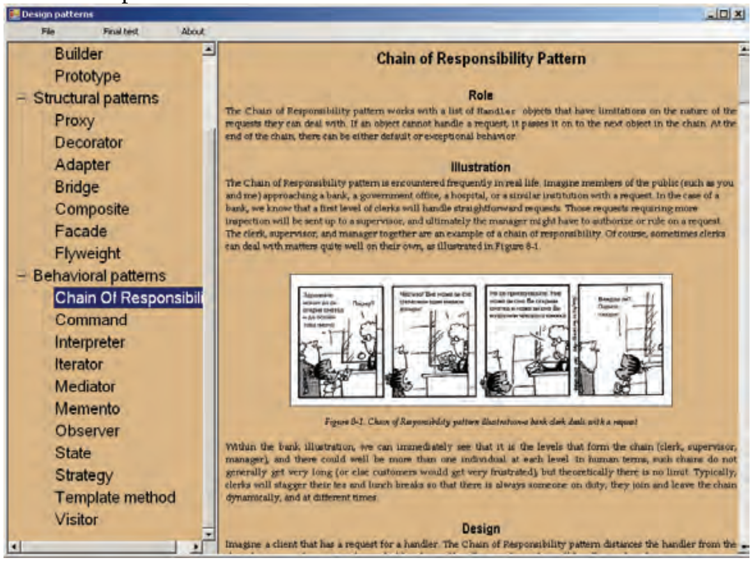
C. Our approach to self-testing and examination of knowledge

Module "Tests" is used by the students to self-checking of learning. There are different ways of organizing test questions. Our module randomly generates a sequence of questions ("close" or "open"), which answers can be given in one of the following ways:

a) For "close" question: By selecting an option (or options) from a list of answers (menu), in which each question provides a list of correct and incorrect answers (figure 3). The student selects one or several answers. The system displays the correct answer but does not provide an assessment of the learner. In the wrong answer to this type of question, the system provides the both the correct answer and the wrong answer (figure 5) - displayed in red.

b) For "open" question: Response in the form of text (figure 4). The system displays the correct answer in the form of "reference text" but does not provide an assessment of the learner (figure 6). The ability to see the correct answers gives the learner a real idea of the extent of assimilation of the material.

Students can go back and change answers to previous question before the test is completed.



Chain of Responsibility Pattern

Role

The Chain of Responsibility pattern works with a list of Handler objects that have limitations on the nature of the requests they can deal with. If an object cannot handle a request, it passes it on to the next object in the chain. At the end of the chain, there can be either default or exceptional behavior.

Illustration

The Chain of Responsibility pattern is encountered frequently in real life. Imagine members of the public (such as you and me) approaching a bank, a government office, a hospital, or a similar institution with a request. In the case of a bank, we know that a first level of clerks will handle straightforward requests. Those requests requiring more inspection will be sent up to a supervisor, and ultimately the manager might have to authorize or rule on a request. The clerk, supervisor, and manager together are an example of a chain of responsibility. Of course, sometimes clerks can deal with matters quite well on their own, as illustrated in Figure 8-1.

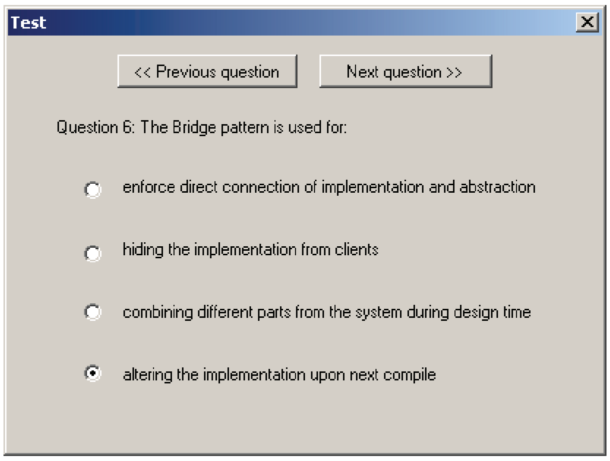
Figure 8-1. Chain of Responsibility pattern illustration—a bank clerk deals with a request

Within the bank illustration, we can immediately see that it is the levels that form the chain (clerk, supervisor, manager), and there could well be more than one individual at each level. In human terms, such chains do not generally get very long (or else customers would get very frustrated), but theoretically there is no limit. Typically, clerks will stagger their tea and lunch breaks so that there is always someone on duty; they join and leave the chain dynamically, and at different times.

Design

Imagine a client that has a request for a handler. The Chain of Responsibility pattern distances the handler from the

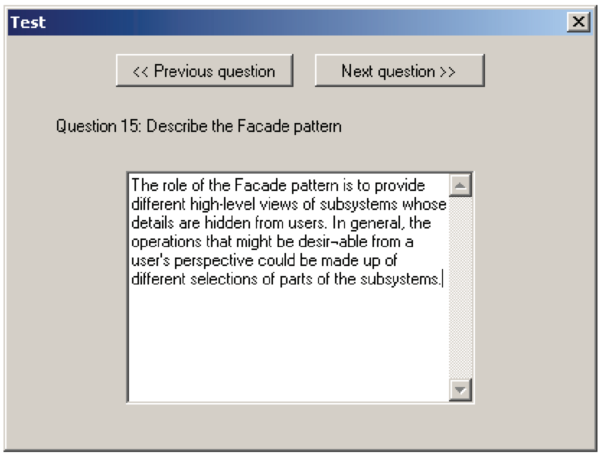
Fig. 2.The main window of the electronic textbook



Question 6: The Bridge pattern is used for:

- enforce direct connection of implementation and abstraction
- hiding the implementation from clients
- combining different parts from the system during design time
- [✓ altering the implementation upon next compile]

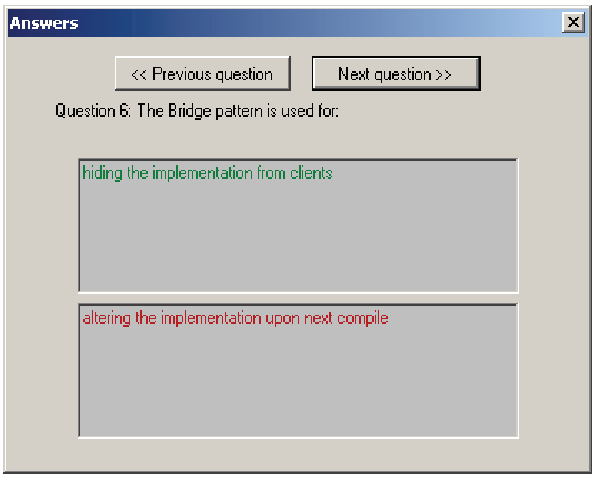
Fig.3. A “close” test question



Question 15: Describe the Facade pattern

The role of the Facade pattern is to provide different high-level views of subsystems whose details are hidden from users. In general, the operations that might be desir¬able from a user's perspective could be made up of different selections of parts of the subsystems.

Fig.4. An “open” test question

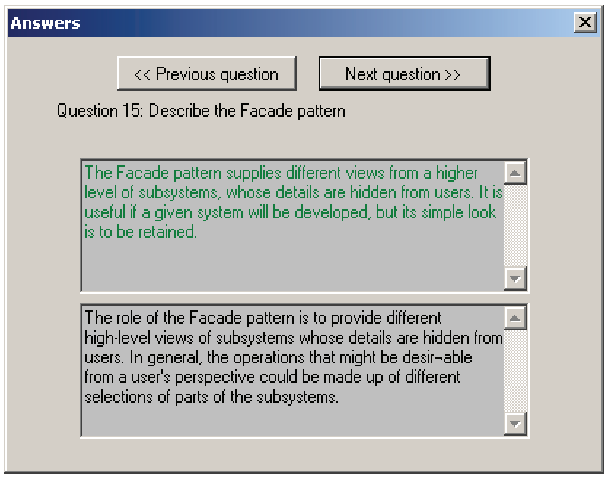


Question 6: The Bridge pattern is used for:

hiding the implementation from clients

altering the implementation upon next compile

Fig. 5. A “close” question answer



Question 15: Describe the Facade pattern

The Facade pattern supplies different views from a higher level of subsystems, whose details are hidden from users. It is useful if a given system will be developed, but its simple look is to be retained.

The role of the Facade pattern is to provide different high-level views of subsystems whose details are hidden from users. In general, the operations that might be desir¬able from a user's perspective could be made up of different selections of parts of the subsystems.

Fig. 6. An “open” question answer

III. Conclusion

In this paper, we discuss some problems of teaching “Software Design Patterns” in the master's degree education of specialty Computer Systems and Technologies in the Technical University of Varna. Based on our experience and observations in teaching similar courses we could claim that the discipline "Software Design Patterns" is undoubtedly useful for students in master's degree. It is because design patterns are elegant solutions to typical problems in the software design, with the possibility of reuse. They are a way to increase efficiency and quality of programming work in developing advanced software applications.

Next, the paper presents our approach to teaching „Software Design Patterns" - an approach based on the method of an electronic textbook. The paper highlights the traditional and the specific features of the developed electronic textbook and comments its structure, its implementation and mainly - the approach used for self-testing and student’s knowledge examination. We could also argue that the proposed teaching approach for the course "Software Design Patterns" is innovative and effective. This is because it is computer based, i.e. it is based on the latest teaching methodology both to the learning process, and also to students who are trained. The created textbook on the subject "Software Design Patterns" contains material for various levels of complexity, it is much more compact than traditional printed textbooks (it is collected in one CD) and it is accessible from any workstation (it is installed on each computer). It focuses on practical examples instead of theory. Based on our more than twenty years practice in Computer Sciences and Engineering Department of the Technical University in Varna we are persuaded that accentuating on practical examples instead of theory is the best way of motivating the students to use some technology. It provides personalized information space for each student and much higher visibility than the traditional printed textbook. It provides a variety of tests with varying degrees of complexity in interactive teaching mode; there is a feedback - in an incorrect response from the student, the system offers the right answer.

There are many directions for development and improvement of the textbook developed, although it complies with both the traditional and the specific requirements. Let mention only some of them: adding audio, video fragments and animations in MAIN PART; adding new software patterns; adding of recommendations and examples to assist students when implementing the stated laboratory tasks; transition from Desktop implementation to WWW-based application. There are also many opportunities to improve the testing module: firstly, the e-book has to enable the teacher to control the absorption of knowledge, next - implementation of a system for assessing the acquired knowledge is needed. The evaluation of the students' knowledge for each “close” question could be realized using for example the formula, given in [4]: $y = r / (N + f)$, where r is the number of correctly selected items from the list of responses, N - number of correct answers in the list, f - number of incorrect items from the list of answers. Criterion for the correctness of the answer for an open question might be the presence of a number of key words in the reference text

All these ideas for further development aimed at achieving even greater visibility of the presented teaching material, even greater completeness of the contents of the course and accordingly: higher efficiency of the learning process.

ACKNOWLEDGEMENT

This work is supported by a scientific research project in Technical University of Varna.

REFERENCES

[1] Shirley Tessler, Avron Barr, Nagy Hanna, National Software Industry Development: Considerations for Government Planners, http://www.aldo.com/Publications/Papers/National_SWI_Development_050303.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, http://www.uml.org.cn/c++/pdf/DesignPatterns.pdf

[3] E-book http://en.wikipedia.org/wiki/E-book

[4] Норенков И.П., Информационные технологии в образовании, http://bigor.bmstu.ru/?cnt/?doc=Default/050_iteduc.cou

[5] C# 3.0 *Design Patterns*, Judith Bishop, O’Reilly Media, 2008