Programming Case Studies as Context for Active Learning Activities in the Classroom

Yonglei Tao
College of Engineering and Computing
Grand Valley State University
Allendale, MI 49301, USA
taoy@gvsu.edu

Jagadeesh Nandigam
College of Engineering and Computing
Grand Valley State University
Allendale, MI 49301, USA
nandigaj@gvsu.edu

***Abstract* – Programming studies are used to achieve a variety of instructional objectives in computer science courses. Instead of using a programming case study as the lecture material, we adapt it to facilitate active learning activities in the classroom. While it preserves most programming tasks in context, our approach provides an opportunity for students to work out desirable solutions as they learn programming concepts and techniques. An anonymous student survey indicates that students' satisfaction with their learning experience was noticeably enhanced. In this paper, we describe our initial experience with a case study-based approach to active learning. We also discuss issues that we have identified with this approach.**

***Keywords* – Active Learning, Programming Case Studies, Computer Science Education.**

I. INTRODUCTION

Many students are more inductive than deductive reasoners; they can learn better from specific examples than from logical development starting with general principles. Hence, the use of case studies is an effective classroom technique [2].

Programming case studies have been used in computer science courses to assist instructors in achieving their instructional objectives through in-class lectures. Some of them also serve as a basis for wide-ranging exploration and in-depth discussion on issues such as problem solving strategies and programming techniques. Moreover, separate assignments are often given as a way for students to apply what they have learned. However, such an approach does not always work effectively in engaging students with diverse backgrounds and varied inclinations in a programming class.

Instead of using a programming case study as the lecture material, we adapt it to facilitate active learning activities in the classroom. While preserving most programming tasks in context, we create an opportunity for students to work out desirable solutions as they learn relevant programming concepts and techniques. We believe that a shift from an instructor-centered approach to a student-centered active learning approach can better engage students in learning.

We conducted an experiment with our case study-based approach to active learning. An anonymous student survey reveals that students' satisfaction with their learning experience was noticeably enhanced. Students were motivated due to the fact that they had to learn what they needed to use in order to solve a meaningful problem. Many of them felt that they had learned more than they would in the conventional way.

This paper is organized as follows. Section 2 covers background and related work. Section 3 introduces our case study-based approach to active learning and section 4 gives an example of using this approach in a CS course. Section 5 describes the results of a student survey. Section 6 discusses our initial experience as well as reflection on this approach. Finally section 7 concludes this paper.

II. BACKGROUND AND RELATED WORK

Bonwell and Eison define active learning as anything that *involves students in doing things and thinking about the things they are doing* [5]. General characteristics commonly associated with active learning are: students are engaged in more activities than just listening; more emphasis is placed on developing students' skills than on transmitting information; students are engaged in activities such as dialog, debate, reading, writing, and problem-solving; students are involved in higher-order thinking (analysis, synthesis, evaluation) activities; and greater emphasis is placed on students' own exploration of subject matter.

Active learning can be incorporated into classroom using various activities. A spectrum of some active learning activities arranged by complexity and classroom time commitment, along with brief descriptions for each of the activities on the continuum, is summarized in [6].

Case studies can be used as an effective mechanism for incorporating and promoting active learning in the classroom. Although case studies have long been used extensively in the teaching of business, medicine, law, and social sciences, they can be used in any discipline where a chunk of realistic problem/story can be brought into the classroom to be worked over by the students and the instructor [2, 7]. Linn and Clancy make the case for using case studies in programming by

comparing the use of case studies to apprenticeship learning in the domain of programming [10]. Their results indicate that case studies are effective for communicating complex, multi-dimensional decisions necessary for designing an effective computer program.

Some example areas of using case studies to enable active learning in computer science discipline include object-oriented programming and design patterns [14, 15], software engineering and project management [9, 12], usability engineering [8, 13], and operating systems [11].

While most proposals in this regard use case studies as they are written to achieve a variety of instructional objectives, our approach is to adapt a case study so as to allow students to work out a solution in the original context as they learn programming concepts and techniques.

III. OUR APPROACH

A programming case study usually consists of a problem statement, development strategies, and one or more solutions, including source code and a narrative description as its documentation, and optionally tool support involved. Most case studies are either based on an actual application, or are a construction of a reasonable situation in which common design issues are identified.

Programming case studies are widely available in the literature; many of them are well-written and proven to be successful in meeting instructional objectives, especially those that are included in a number of popular textbooks. While they are readily usable, the challenge for instructors is how to teach programming with a case study.

Our approach is to adapt a programming case study to support student-centered active learning activities in the classroom. Adaptation creates a case study with a partial solution (including unfinished source code). Students make collective effort to complete the work as they learn programming concepts and techniques. The key elements of our approach are described below.

We select a programming case study according to our instructional objectives and identify relevant programming tasks from the available solution. Our goal is to discover tasks that not only are meaningful in the given context, but also impose reasonable challenges to our students. We then remove statements that correspond to those tasks from the given source code and supplement them with program comments to describe the missing functionality. For example, if the functionality of a self-contained program unit, such as a method of a class in an object-oriented program, is chosen as a relevant task, then statements in the body of that method is replaced with comments about its pre and post conditions. Here, program comments are added to provide instructions for students to find what they need to do.

Some programming tasks that we find relevant to our purposes are quite difficult for students to complete within the time frame allowable in the classroom. Even if it is possible for students to do those tasks in their entirety, it would divert too much of the valuable class time away from more important issues. Hence, we introduce helper methods to hide complex processing logic that students need to use but are not ready to tackle. Students use helper methods as building blocks to perform their tasks. Using helper methods makes our programming tasks practically feasible as activities in the classroom.

A case study tells a story about user needs in a realistic situation and also describes a process to develop a program to meet the needs. In Agile development methods, a story is considered as a unit of work that the developer carries out in a development cycle. It is beneficial to give students a sense of participation as a team in such a development cycle. If they do not have adequate understanding about the context, students may lose sight of the forest for the trees. Hence, we organize programming activities in an “outside-in”, rather “bottom-up”, manner [4]. Using an “outside-in” strategy allows students to focus on satisfying the needs identified in the given context, and subsequently helps them cope with the complexity of their programming tasks.

As a consequence, adaptation of a case study creates a partial solution to a practical problem with several programming tasks for students to accomplish. We believe that the focus should be placed not only on writing code, but also on the process of a team-based development cycle. It is crucial to have students form groups to analyze their tasks and work out a solution before the class discussion takes place. Since much of the case study remains intact, it provides a meaningful context for students to learn programming concepts as they make collective effort to complete the partial solution to the case study.

IV. AN EXAMPLE

We have conducted an experiment with the case study-based approach discussed above in a class that was intended to teach object-oriented programming in Java and prepare students for their graduate study. Students in this class had diverse backgrounds and their prior knowledge about programming varied greatly; what they knew about programming was not necessarily Java and what they wanted to pursue as an emphasis of their graduate study was very different. Our experience from the past offerings of this course indicates that the lecture-based approach does not always work effectively for students in this class.

We selected a programming case study from [1] in order to teach how to write a single class in an object-oriented program. In the original case study, the case of creating a user-defined class, named as Day, is built around the needs for processing dates in a variety of situations in which a total order on dates is an essential consideration (a primary reason behind the design decision of creating a user defined class, rather than using a built-in class in Java). In addition, programming concepts and principles, such as encapsulation, software quality attributes, programming by contract, and unit testing, are deliberated. We have developed three class exercises based on this case study, making it possible for students to carry out active learning activities in the classroom.

Class exercise one is to explore the public interface of class Day. Students are asked to write code using methods of this class to perform required functionality, such as finding the

number of days between two specific days and determining the day that is a number of days away from a given day. Students also need to discover different ways to accomplish certain objectives, including taking a direct action on a day object that is returned by a prior action. This class exercise helps students obtain adequate understanding of the observable behavior of a class, and also paves the way for them to write unit tests before writing the code as practiced by Test-Driven Development (unit testing is a separate but related topic in the class).

Class exercises two and three are to implement class Day in different ways. While exercise two represents the state of a Day object as year, month, and day, exercise three represents it as a single Julian day number. Students are asked to write code to implement the public methods of class Day. Certain functionality is quite overwhelming for them, for example, finding the next day or the previous day for a given day in exercise two as well as converting between a Julian day and a calendar day in exercise three. Helper methods are made available for students to use in order to reduce the complexity of their programming tasks, but the processing logic of these helper methods is hidden from students completely. In addition, we have added comments to provide instructions about what they need to do.

Class exercises as such comprise a partial solution in the original context. They serve as the basis for active learning activities for students. We also provided guidance for students to work out a complete solution in an incremental manner. In each development phase, students worked on one exercise in groups. Group discussion allowed them to learn what they need to know and collaboration made it possible for them to complete what they need to do. Then students took part in a follow-up class discussion to share their code, inquire into certain issues, and determine a desirable solution. At the end of each development phase, the instructor summarized what had been accomplished and allowed students to move on.

V. EVALUATION SURVEY

We have conducted an anonymous survey and analyzed feedback from students in the foundation-level graduate class described above. The class had 30 students enrolled during the fall semester last year and 70% of them participated in the survey. Typical students enrolled in this class are part-time working professionals and full-time (mostly international) students with varying backgrounds and preparedness in programming and computing, in general.

Analysis of the student survey revealed that most students considered the learning experiences they had through the use of case study-based exercises as the most beneficial. Obviously, students were motivated to acquire new knowledge. Listed below are a few comments from students:

- "I like the flipped class room approach. I thought it was very effective."
- "They were an excellent tool and contributed to my learning in this class."
- "Exercises completed in class together were stimulating. They accurately represented what we were learning in practice."

Students also considered that they had learned more than they would in the conventional way. They viewed the knowledge and experience they gained from active learning activities as being valuable. Several comments from students are given below:

- "That was the best part of class where we could test ourselves and learn how our fellow programmers were thinking."
- "I felt that the in-class exercises really helped to have us all work on a program together and the instructor could address issues that may have come up."
- "Literally every time I would work on a project or on homework, I would consult the in-class work as a reference, example, and guide. I even wound up using them to help guide my thinking through projects for myself as well as other classes I am currently enrolled in."

Students also provided feedback and suggestions on how to improve the course offering next time using the case study-based approach. They reminded us of paying attention to those students who were less prepared for a class like this. They also suggested ways of making our class exercises more effective. Some examples of student suggestions are listed below:

- "Keep the in-class exercises based on class composition."
- "Good, but this course should probably be thought in a lab where students (especially those new to Java) can get hands on experience."
- "But just that the intensity could be reduced. Either less number of coding exercises or less intensive."

Clearly, students appreciated the value of using case study-based in-class exercises in learning object-oriented programming. These exercises had the students actively engaged in classroom, making learning an active rather than a passive process. There is definitely room for improvement in how we use the case study-based approach to effectively teach the subject matter to students coming from wide educational backgrounds and work experiences.

VI. DISCUSSION

Programming case studies provide a realistic situation to encourage just-in-time learning of new knowledge. It makes it less compulsory for the instructor to motivate students. As a consequence, the emphasis of teaching has shifted from preparing students for what they need to know to creating opportunities for them to discover what they need to learn.

A partial solution that we created in the context of a programming case study is intended to support student-centered classroom activities. Due to time constraints, students need clear instructions on what their responsibilities are; ideally, they should be able to depend entirely on the information present in the partial solution and do not conduct extra research to carry out their work. Moreover, it is helpful for the instructor to give a brief introduction about the case study under consideration before turning over the responsibility to students.

When learning on the basis of a case study, students have to explore an unfamiliar domain and may run into unexpected obstacles. Our experience with this approach indicates that

guidance from the instructor is still beneficial. In general, the instructor should act as a facilitator, rather than as an observer. When students work in groups according the instructions, the instructor may walk around to inspect progress each group has made, answer questions students raise, and intervene, if necessary, in case a group is unable to catch up with other groups. Our experience shows that readily available assistance from the instructor can ensure a smooth development process. However, it may be problematic for a large class.

It is especially helpful for the instructor to participate in class discussion. Students tend to focus on the solution that occur to them right away, rather than thinking about alternative ways to accomplish the same objective or considering issues involved in the development process. In such a situation, the instructor can help them broaden their views and acquire knowledge that they might not realize they need to know.

Styles of learning vary from student to student. Our case studies are programming intensive. Using such a case study in the classroom did not seem to benefit all students equally. For example, writing code in a time-constrained setting was not what many students have been accustomed to in school and it was especially tough for those students who were not well prepared for such a class. In group work, students helped each other by sharing their personal knowledge and experience, which makes it possible for students to move through the learning process together. Still there were students who were noticeably less active than other members in their groups. In order to accommodate students who need more time to figure out what they need to do, a solution under consideration is to allow students to complete an exercise over two consecutive class meetings with a sufficient time span in between.

A few students wanted to make our class exercises a project that they could work on with an IDE (Integrated Development Environment). We have debated about whether to have students work in a laboratory environment. A laboratory environment makes it possible to integrate relevant topics, such as unit testing, into a programming case study. On the other hand, however, the classroom setting allows students to focus on concepts and solutions, rather than syntactic issues, which is more beneficial in their learning when time constraints are a primary concern.

VII. SUMMARY

As technologies evolve, developers in the future are more likely to work with systems that they cannot possibly develop from scratch or use diverse tools that they were not taught in their former education [3]. It is beneficial for students to gain experience with programming in a non-traditional way, especially using, modifying, and expanding the functionality of available components, either as whole or in part, to build useful products.

Programming case studies provide a rich basis for the instructor to organize and bring to life abstract and disparate concepts. Most of them are amendable to serving various instructional objectives. Using our case study-based approach can help students take much more responsibility for their own learning. By placing students in realistic situations and asking them to address programming issues, case studies enable students to connect their knowledge with the application of practical skills, and therefore engaging them in their learning.

Based on our initial experience, we believe a case study-based approach to active learning will find its place as a way to help students efficiently master programming concepts as well as practice program design principles. We plan to continue our effort of incorporating this approach into our programming courses and refining it as we gain more experience with it. We also plan to investigate pedagogical issues that we have identified, including how to help those students who are less prepared to tackle the in-class exercises assigned to them as well as whether to perform case study-based active learning activities in a laboratory environment.

REFERENCES

[1] C. Horstman, Object-Oriented Design & Patterns, 2nd editiion, John Wiley & Sons, Inc., 2006.

[2] Center for Teaching and Learning, "Using Case Studies to Teach," University of Buffalo, http://www.bu.edu/ctl/teaching-resources/using-case-studies-to-teach/.

[3] J. Cohen, "Updating Computer Science Education," Communication of ACM, Vol. 48, No. 6, June 2005, pp. 29-31.

[4] C. Kessler and J. Sweitzer, "Outside-in Software Development: A Practical Approach to Building Successful Stakeholder-based Products," IBM Press, 1st Edition, 2008, PP. 1-3, 23-31, 183-195.

[5] C. C. Bonwell and J. A. Eison, "Active Learning: Creating Excitement in the Classroom," ASHE-ERIC Higher Education Report No. 1. The George Washington University, School of Education and Human Development, Washington, DC., 1991, http://files.eric.ed.gov/fulltext/ED336049.pdf

[6] Center for Research on Learning and Teaching, "Active Learning Continuum," The University of Michigan, http://apcentral.collegeboard.com/apc/public/courses/teachers_corner/151155.html.

[7] Eberly Center of Teaching Excellence and Teaching Innovation, "Instructional Strategies – Case Studies," Carnegie Mellon University, http://www.cmu.edu/teaching/designteach/teach/instructionalstrategies/casestudies.html.

[8] J. M. Carroll and M. B. Rosson, "A Case Library for Teaching Usability Engineering: Design Rationale, Development, and Classroom Experience," ACM Journal on Educational Resources in Computing, Vol. 5, No. 1, March 2005.

[9] S. Kurkovsky, "Teaching Software Engineering with LEGO Serious Play," ITiCSE'15, July 2015, Vilnius, Lithuania.

[10] M. C. Linn and M. J. Clancy, "The Case for Case Studies of Programming Problems," Communications of the ACM, Vol. 35, No. 3, March 1992.

[11] C. R. G. Helps and S. A. Renshaw, "Design of a Flexible Case-Study Instructional Module for Operating Systems for Information Technology," SIGITE'04, October 2004, Salt Lake City, Utah, USA.

[12] P. M. Papadopoulos, S. N. Demetriadis, and I. G. Stamelos, "Case-Based Instruction on the Web for Teaching Software Project Management," ITiCSE'07, June 2007, Dundee, Scotland, United Kingdom.

[13] M. B. Rosson, J. M. Carroll, and C. M. Rodi, "Case Studies for Teaching Usability Engineering," SIGCSE'04, March 2004, Norfolk, Virginia, USA.

[14] College Board AP Central, "The GridWorld Case Study," http://apcentral.collegeboard.com/apc/public/courses/teachers_corner/151155.html.

[15] C. Nevison and B. Wells, "Using a Maze Case Study to Teach Object-Oriented Programming and Design Patterns," Sixth Australasian Computing Education Conferene (ACE2004), Dunedin, New Zealand.