Improving Programs' Quality in Programming Courses

Syed M. Rahman Paul L. Juell
Department of Computer Science, North Dakota State University
258 IACC Building, Fargo, North Dakota 58105, USA
{Syed.Rahman, Paul.Juell}@ndsu.edu

Abstract

In our research, we developed a new software development method by emphasizing software-testing phase to improve programs' quality. Our method has been implemented in last five semesters in the Department of Computer Science at North Dakota State University. This method improved students' program quality at least 24% (in terms of black box testing) comparing to other sections that did not apply this method. Students claimed that this method helped them understanding the problems and writing the codes, made the code easier to debug, helped them debugging their codes, and improved the students' code reliability and quality. In this paper, we have also recommended a few additional activities that we believe need to consider teaching introductory programming courses so that students could produce higher quality programs.

1 Introduction

Many researchers found that computer science (CS) students' in introductory programming courses that completed CS-1 and CS-2 or even who completed the degree produce the code of low quality. An industry survey has reported that more than 50% of a software project's budget is spent on activities related to improving software quality. Industry leaders claimed that this is caused by the inadequate attention paid to software quality in the development phase. The classroom experience shows that students in programming classes produced "toy" programs that are superficially tested, graded, and eventually discarded. Generally, students are not worried about the quality, reuse, or maintenance of their programs. We believe our existing teaching style of programming courses does not prepared students producing higher quality programs.

In our research, we developed a new software development method by emphasizing software testing phase and it has been implemented in last five semesters in the Department of Computer Science at North Dakota State University in five different introductory programming courses. This method improves students' program quality at least 24% (in terms of black box testing) comparing other sections that do not apply this method. Students reported that this method helped them understanding the problems and writing the codes, made the code easier to debug, and improved the students' code reliability and quality.

Besides applying the new software development method for teaching introductory programming courses to improve students' programs quality, we have recommended a few other activities that need to be considered in teaching undergraduate CS programming or related courses; however, our research couldn't provide any quantitative measurement of the effectiveness for the following features:

a) **More testing need to teach in the undergraduate courses in general.** CS or related majors in the undergraduate course curriculum should emphasize on software testing, as it is the most expensive (typically 40-70% of the total cost) phase in software lifecycles both in terms of money and time.
b) **Provide feedback or comments online if the instructor post scores online**. Most of the students do not pickup their assignments or projects and check the mistakes they made. Most likely they would make the same mistakes in the later project or assignment.
c) **Programming courses should have at least two to three group projects** so that students get work experience in a group and gather many other skills such as how to distribute the work in a group into pieces, how to integrate those pieces, brainstorming, and learn from each other.
d) **Identify the most common mistakes or weaknesses and instructors emphasize teaching those topics**. Instructor can identify the common mistakes and deliver more lectures or emphasize teaching those items.
e) **More lab classes are useful for learning computer programming** and in lab classes should have teaching facilities so that instructor can teach in the lab class and students can follow the instructor.

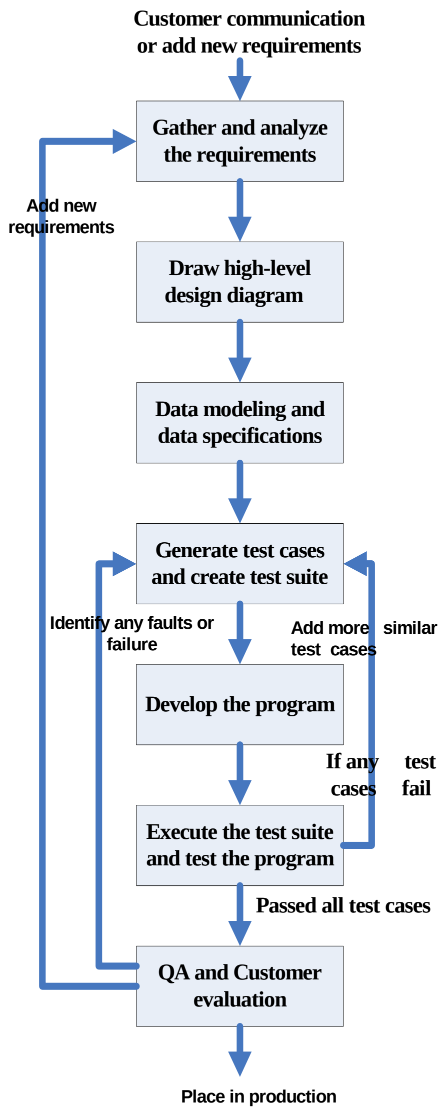
f) **Improve students' participations in the classroom**. Increase the students' participation in the classroom. Students' attention can be maintained throughout a class session by periodically involving students' in some activities.

2 Apply the Testing Before Coding Method for Teaching Courses

While teaching we have witnessed students in the programming courses write toy programs that are superficially tested, graded, and then eventually discarded. Students do not focus on code reuse, integration, quality, or maintenance of their programs. Depending on software type typically, software testing is about 40 % to 70% of total cost of software in the industry although usually testing receives very little attention in CS curriculum [4][9]. Testing is the single most expensive phase in the software lifecycles; however, software failure cost is still very high. National Institute of Standards & Technology released a report on June 28, 2002 that software failures cost the US economy an estimated $59.5 billion per year [3]. We believe one of the root causes of this problem is our educational system where students are not received enough background or training how to develop higher quality software or even how to test their own code properly. They do not learn how to integrate, test, reuse, or maintenance their codes.

In our research, we have addressed this problem and introduced a new software development method that emphasizes software-testing phases. Our method makes a cultural change in software development, and produces higher quality programs. Our method, **T**esting **B**efore **C**oding (we refer to it as **TBC**) does not require previous background in applying software lifecycles and can be adopted in the existing programming or related courses without changing the course contents, syllabus, policies, or course loads.

The TBC method has been applied in teaching undergraduate computer programming courses for the last five semesters in the Department of Computer Science at North Dakota State University in five different introductory programming courses. More than 150 students in each semester participated in our experiment, and the sections that applied the TBC method improved at least 24 % of their programs' quality (in terms of black-box testing) comparing to the sections that did not apply the TBC method. Our experiment also concluded that applying the TBC method in program development helped in understanding the problems and writing the code, made it easier to do debugging, improved the quality and reliability of the codes, boosted developers' confidence, and so on.



Customer communication or add new requirements
Gather and analyze the requirements
Add new requirements
Draw high-level design diagram
Data modeling and data specifications
Generate test cases and create test suite
Identify any faults or failure
Add more similar test cases
Develop the program
If any test cases fail
Execute the test suite and test the program
Passed all test cases
QA and Customer evaluation
Place in production

Figure 1: TBC development method

2.1 Evaluations of the TBC Method

Testing before coding method (Figure: 1) is suitable for teaching the introductory programming courses in undergraduate computer science or related majors. We also assume that the TBC method can be adopted in the industry. Because TBC provides wide range of the developers do not need any training, adding or changing the requirements would be cheaper, and so on. In section 2.1.7, we have discussed why industry should adopt the TBC method.

In our experiment, the same instructor teaches the same course in two different sections. One section students' follow the TBC model, generate test suites before writing the codes, and another section's students did not. Keeping all factors unchanged, we found that the TBC model improve the students' program quality. We found that teaching basic concept of software testing does not take much time.

2.1.1 TBC improves the Students' Program Quality

We created a test suite following different testing techniques such as boundary value analysis and equivalence partitioning. We executed all test cases in all students' programs in both sections. Keeping all factors such as textbook, course contents, syllabus, teaching materials unchanged, in both sections, we found that the passing rate of the black box test suite in the TBC section was at least 24% higher than the non-TBC sections.

2.1.2. TBC method can be adopted in the existing programming courses

We understand that introductory programming courses already are in huge load for the students. There are too many things that need to cover in one semester, and there is no room for additional loads. In our experiment, TBC method has been adopted in the existing courses without changing any course contents, syllabus, grading policies, or course loads.

2.1.3. A Cultural Shift in Software Development

In the traditional software development method (e.g. waterfall method), most of the time developers do not test their code properly. In the TBC approach, we make a *cultural shift* in teaching programming languages by making testing an integral part of programming practices and apply software engineering concepts in programming courses. In our approach, developers or students must learn how to test their own code. Not only producing the *correct output* or a *compiled* version of the code, in our opinion, if developers or students know how to write the program then they better know how to test it and make sure that their programs are doing what they expected it to do.

2.1.4. Makes the program easier for debugging and improves the code's reliability and quality

In the TBC method, students write a test suite before writing the codes. Students execute the whole test suite and record the results and find out if there are any faults and failures in the programs. If any test case fails, students would know exactly which test case fails and fixes the corresponding codes. They will also generate a few additional similar test cases to investigate properly and execute the whole test suite again to make sure that their program does not break somewhere else for the recent changes.

2.1.5. Spends extra time upfront and these extra efforts eventually well payoff in program development.

TBC method forces the developers to understand the problem better while they specify the requirements, specify the data, and write test cases before coding. Developers or students get advantages while implementing the program. In our experiment, we surveyed the total time spent on few specific projects/assignments in both the sections who applied the TBC method or who did not apply the TBC for developing their programs. We could not find any significant difference in total time spent on the program development.

2.1.6. Increases developers' confidence in the correctness of their programs

In the TBC method, students follow the instructions and several steps/cycles to develop the program. Students generate test cases and create a test suite. They execute the program and execute the test suite to test their programs. If the program passes all test cases, it increases the confidence of the students about their programs' correctness. The developers' confidence also increases while making any changes in the program. After making any changes, the developers execute the test suite and make sure that the recent change in one place does not break the program somewhere else. Even if it does, the developer would know the exact reason and place to fix the problem.

2.1.7. Apply TBC method in the software industry

TBC method is not only suitable for the introductory programming courses but also in the software industry. To some extent the developers in industry are not much different from the students. An industry survey reported [7] that more than 50% of a software project's budget was spent on activities related to improving software quality, and the survey stated the developers did not put enough attention in quality while developing the program as the reason. The developers in the industry would get the benefits that we discussed in the above section. We believe the TBC method would be suitable in the industry because:

- TBC method is very useful for understanding the requirements and writing the programs.
- Experience developers can adapt the TBC model quickly and produce higher quality programs. Non-experience or new developers can start developing software soon, as TBC does not require any training.
- TBC makes the program easier to find errors in the developers' code.
- It increases the developers' confidence in the correctness of their programs and encourages them to move forward.
- It makes the program easier to maintain and modify and is less expensive for changing the requirements or adding any new requirements.
- TBC reduces the software testing cycle and save huge money in the industries, as the testing cycle would be very short in the TBC method.

It is easy to manage the project and easier for customer communication in the TBC method. Because everyone works closely in the project and maintain a constant communication that would lead the project in the right direction.

3 Some Other Important Activities for Improving Students' Program Quality

We have addressed several activities or issues that we believe need to take to improve students' program quality. We have made the following recommendations so that students' would produce better quality programs and would prepare to face the real world's challenges in this field.

a) **More testing need to teach in the undergraduate courses in general.** Depending on the software, 40-70% of the total cost is only for software testing phase. However, CS or related majors in the undergraduate course curriculum does emphasize on software testing. Most of the faculty and staff in the survey claimed that approximately 60% of the undergraduate students in the CS dept do NOT know how to properly test their code. We believe that we should teach at least one software testing course in both undergraduate and graduate level students. Students should learn and practice software testing from the beginning of their programming courses.

b) **Provide feedback or comments online if the instructor post scores online**. In our research, faculty and staff survey found that at least 50% of the students do not pick up their assignments or project as long as they receive the score online. If they do not pick the assignment most likely they would make the same mistakes in the later project or assignment.

- ◈ **Students should receive feedback online if they receive their score online.**
    - o If students submit their work online using a digital drop box, email or any other web application, they should receive the feedback when they would see their scores.
    - o This can be done through a web application or even instructor could create a excel datasheet and upload in the course website with comments / feedbacks. Instructor could provide a separate secret password to every student so that they would be able to check their score and feedback.

- ◈ **Automated Grading Tool**
    - o We found that most of the students do not pick their programming assignments and projects. They do not learn from their mistakes and end up making the same mistakes repeatedly. We recommend using an automated tool where students would submit their assignments or projects.
    - o The tool would evaluate students' work and provide an immediate feedback to the students.
    - o Students would upload their work online and verify their work several times before final submission deadline.
    - o The tool would receive students' generated test cases as inputs. Students should get partial credit even if their code would not compile. The tool should not only look for correct output and compiled version of the code.

c) **Programming courses should have at least 2 to 3 group projects** so that students get work experience in a group and gather many other skills such how to distribute the work in a group into pieces, how to integrate those pieces, brainstorming, and learn from each other.

- ◈ **Assign more group projects in the programming courses**
    - o Students learn more from each other. Students feel more comfortable asking question or sharing ideas to the group members than discus with the instructors. Most importantly, instructors are not accessible like group members.
    - o Students earn valuable group-work experiences and receive a real world work flavor.

d) **Identify the most common weaknesses and instructors emphasize teaching those topics**. Identify the common mistakes and weaknesses of the students and instructors would spend more time or explain in those topics.

- **Instructor should emphasis on teaching students’ weaker topics**
    - The instructor identifies the students’ weaknesses and emphasis on teaching those topics.
    - The grader or teaching assistant who grades the courses make a list for the common mistakes to the instructor and the instructor could spend few minutes discussing in the classrooms.

- **Instructors need to communicate with students’, graders, or TA’s to find out the students’ weaknesses.**
    - Instructors need to oversee the graders or TA’s work so that he/she would have an idea about students’ common mistakes and discus in the classroom.
    - Grader could provide a list of problems/weaknesses that students would have in that class.

e) **More lab classes are useful for learning computer programming**. Most of the faculty & staffs and students in different programming courses expressed that in the introductory programming classes, the ratio of the classroom lecture to lab sessions should be at least 50% or more labs. Lab classes should have teaching facilities so that instructor can teach in the lab class and students can follow him.

f) **Improve students’ participations in the classroom**. Increase the students’ participation in the classroom. Many research found that (such as [2]) most students couldn’t stay focus throughout a lecture. After about 10 minutes their attention begins to drift, first for brief moments and then for longer intervals, and by the end of the lecture students’ take in very little and retain less. A classroom research study showed that immediately after lectures students recalled 70% of the information presented in the first ten minutes and only 20% of that from the last ten minutes [2].

Students’ attention can be maintained throughout a class session by periodically involving students’ in some activities. Many different activities can serve this purpose such as asking questions, recalling prior material, giving some small tasks in the classroom as individual work or as a small group work. We are also actively use *Personal Response System* (PRS) system in the classroom and found more students participate in the classroom and they may not participate in the class or raise their hands; however, using the PRS those students participate in the classroom. We found the PRS system as fun learning tool for the students.

4 Results

We have proposed and implemented a new software development method primarily emphasizing on the software testing phases. Our method has been applied in the five different introductory programming courses and improved students’ program quality and makes a cultural shift in the program development. In our opinion, if the students know how to write the programs then they should know how to test it and make sure that their programs are doing what they expected to do.

There are few findings of our research are given below:

- TBC section students' program quality was 24% higher than non-TBC section, in terms of the black box testing.
- We spent only around 25 minutes how to write test cases and 70%-90% students in the different classes came up with test cases instantly.
- Our approach integrates into the existing programming courses without changing the textbook, course contents, materials, policies or the course loads.
- 88% of the students expressed that TBC method was helpful for understanding and writing the program
- 67% students agreed that TBC method boosts students' confidence and it gave them the courage to move forward
- Most of the faculty and staffs claimed that at least 50% of the students do NOT pick up their assignments or projects and do not learn from their mistakes.
- Faculty and staffs also claimed that at least 60% of the CS undergraduates should improve the quality of their programs.
- Most of the faculty and staffs agreed that at least 60% of the CS students do not know how to properly test their code.
- TBC method is not designed for any programming languages specific or level of courses. It has been implemented in different languages with different level of courses such introductory and intermediate java programming, Visual Basic. NET etc. and improve students' programs.

5 Conclusions

Software testing is the most expensive phase in software development but acknowledges very little attention in CS course curriculum. We found most of the cases, the students write poor quality programs and they do not know how to test their programs. Faculty and staffs expressed that at least 60% of the CS undergraduates should improve the quality of their programs. Students are not worried about the program quality, code reuse, integration or maintenance of their programs. We believe the traditional way of teaching introductory programming courses leaves students unprepared for developing reliable and high quality software.

In our research, we addressed this problem and introduced the TBC model that mainly applies software engineering concepts in program development by emphasizing on software testing lifecycle. Applying this *cultural shift* approach, in teaching introductory programming courses, students improve their programs quality, in terms of black box testing. Not only producing *compiled* version of the codes and *correct* outputs; students must test their own codes and make sure that their programs are doing what they expected to do.

The TBC model integrates into the existing programming courses without changing the course policies, contents or loads and it improves the students' program quality. Most of the students claim that the TBC helps students' understanding the requirements and writing the program; makes the code easier to debug and improve their codes' reliability and quality. Students initially spend some extra time well payoffs in later phases in the program development. We have also made several recommendations for teaching introductory programming courses and making some adjustment in the course curriculum so that students would gather experience how to produce higher quality programs and would be ready to accept real world's challenges.

6 References

[1]. Townhidnejad, Hilburn, Software quality: a curriculum postscript?, Technical Symposium on Computer Science Education, Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States Pages: 167 – 171, 2000

[2]. Felder, Richard and Brent, Rebecca; "How to Improve Teaching Quality" Quality Management Journal, 6(2), 9-21, 1999, http://www.ncsu.edu/felder-public/Papers/TQM.htm

[3]. Newman, Micheal, "Software Errors Cost U.S. Economy $59.5 Billion Annually," NIST Assesses Technical Needs of Industry to Improve Software-Testing, June 28, 2002, http://www.nist.gov/public_affairs/releases/n02-10.htm

[4]. Rahman, Syed, Salah, Akram and Others; "Teaching Software Testing in Introductory CS Programming Courses", the Conference of the Midwest Instruction and Computing Symposium, April 8 - 9, 2005, Eau Claire, Wisconsin, USA.

[5]. Edwards, Stephen H.; "Using software testing to move students from trial-and-error to reflection-in-action", Technical Symposium on Computer Science Education, Proceedings of the 35th SIGCSE technical symposium on Computer science education, Norfolk, Virginia, USA, Pages: 26 – 30, 2004

[6]. Wolfgang Zuser, Stefan Heil, Thomas Grechenig Software quality development and assurance in RUP, MSF and XP: a comparative study, ACM SIGSOFT Software Engineering Notes, Proceedings of the third workshop on Software quality 3-WoSQ, Volume 30 Issue 4, May 2005

[7]. Townhidnejad, Hilburn: Software quality: a curriculum postscript?, Technical Symposium on Computer Science Education, Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States Pages: 167 – 171, 2000

[8]. Pressman, Roger, S; Software Engineering: A Practitioner's Approach, 6th Ed, McGraw Hill, New York, NY, 2005, Page 467-594 Portland, Oregon, 2003

[9]. Rahman, Syed and Salah, Akram; "Teaching Software Testing in Introductory CS Courses and Improving Software Quality", 3rd international Workshop on Modeling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS-2005), May 24, 2005, Miami, FL, USA.

[10]. Rahman, Syed and Salah, Akram; "Adopting Test-Driven Development in Web Applications' Developments" ISCA 20th International Conference on Computers and Their Applications (CATA-2005), March 16-18, 2005,New Orleans, Louisiana, USA

[11]. Beck, Kent; Test-Driven Development: By Example, Pearson Education Inc., 2003