Building software testing skills in undergraduate students using Spiral Model Approach

Gopalkrishna Joshi
Dept. of CSE
K.L.E. Technological University
Hubli, India
ghjoshi@bvb.edu

Padmashree Desai
Dept. of CSE
K.L.E. Technological University
Hubli, India
padmashri@bvb.edu

Abstract—**Spiral Learning aims to strengthen students' understanding of the basic concepts by revisiting the concepts periodically with different contexts and with increasing sophistication throughout the curriculum. This approach helps to overcome the limitations of the instructional design and delivery such as concepts taught in isolation and not being emphasized in later stages of learning leading to poor appreciation of learned concepts and repetition disregarding earlier knowledge limiting the depth of treatment. Software testing is an important skill required for computer science and engineering professionals. Students are able to write quality programs only if they know what is to be tested. Earlier, "software testing" was taught as a course in higher semester of undergraduate (UG) curriculum of computer science and engineering. As it was taught in isolation it was difficult for to build up required testing skill in the students.**

This paper discusses the experience of authors in building software testing skills among the students of engineering degree program in computer science and engineering as a solution to make up for the deficiency in the earlier curriculum. Authors explain the need for building software testing skills seamlessly integrating with programming, software engineering and project courses through spiral approach.

Keywords—**skills; testing; software; curriculum; spiral model**

I. INTRODUCTION

Earlier, Computer Science and Engineering B.E. curriculum had "software testing" as one of the course. Students used to learn theoretical concepts in class room and would write manual test cases for a given problem statement. There were no hands on practice on testing the code or program they had written. This created a gap created between the program writing and actual testing. Time required for implementation was more. "Developer says that one can write correct code only if he knows what is to be tested and when to be tested." Industry expects that the student should have ability to write correct program and test it[2]. It believes that the testing phase can be shortened if the defects are removed in earlier stages. Student cannot practice different levels of testing in one course in a single semester. Mastery in testing can be achieved with practice from simple testing to larger and complex code.

Our experience with students learning process indicated that software testing can be taught in laboratory courses at different semesters by identifying important core courses. We identified Programming in C, Data structures, Object oriented programming with JAVA, Software engineering, Mini project, Minor project and Major/Capstone project and applied spiral model to develop testing skills.

A. Why Spiral teaching?

Spiral teaching is a teaching technique that allows learners to construct much from their own learning process. In this spiral model students first learn the basics of course, without worrying about higher level of details[1]. As learning progresses, complex levels of details are introduced, while at the same time they are related to the basics which are reemphasized many times to help them to enter them into long- term memory.

II. PROPOSED METHODOLOGY

Software testing curriculum is designed based on general Spiraling curriculum framework which has mainly following components [3]: Using this approach, the teacher can spiral up or down through the curriculum, depending on the pace of the students' learning as shown in fig 1.

- **Beginning topic:** Basic topics of the course are introduced. The instructor needs to find out from the student what his experience or understanding level is. Think of it as kind of a contemplative stage, because the student is able to think about it and return.
- **Next topic**: Move to a next level of topic. The students aren't made to feel that they didn't "get" the first topic, but that there will a time to revisit it later.
- **Either return to the first topic or explore another**: The instructor either moves to a new topic or goes back to the initial topic and gauges the students' experience and understanding.
- **Continue to build on the students' understanding of the concept**.

We have introduced software testing in B.E. first year at Programming in C. Students have taught about software testing and its importance in software development. Test driven programming as shown in fig 2 is introduced. First level of software testing that is unit testing[5] is taught.

In Second year at Data structure course and Object oriented programming with JAVA unit testing is revisited and performed component testing[5] for little more complex modules. Students could get better understanding of unit testing. Also integration testing is introduced to connect different modules. However, students got less experience in

978-1-5090-6115-0/16 $31.00 © 2016 IEEE
DOI 10.1109/T4E.2016.60

integration testing and hence understanding of this level testing was less.

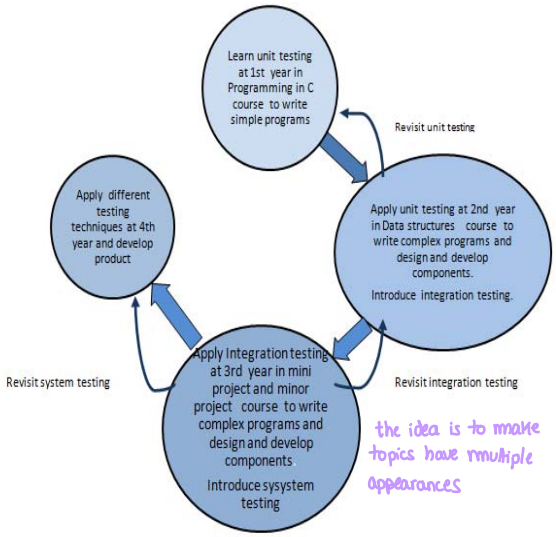

Fig. 1. Spiral model for software testing

In Third year, fifth semester mini project and sixth semester minor project integration testing is revisited and system testing is introduced. Students could test developed system's functionality using system testing. Students gained systematic knowledge on integration testing. Students learnt system testing as new testing technique.

In fourth year at Capstone project students build complex system. They follow the software testing life cycle activities. At the end of fourth year students acquired software testing skills through spiral learning of different testing techniques without much stress.

Different tools and technology are used at different levels such as JUNIT, Selinium and CodePro for testing the modules and review the quality of the code written. This approach helped us to put the theory of thinking into practice in concrete way at the classroom level. Also the "learning in spiral" as a tool developed concrete practical thinking that made students' performances of understanding explicit and visible. Sample test plan done for data structure course is shown in the fig 3. Test plan also includes assumptions and constraints that need to considered in the program design.

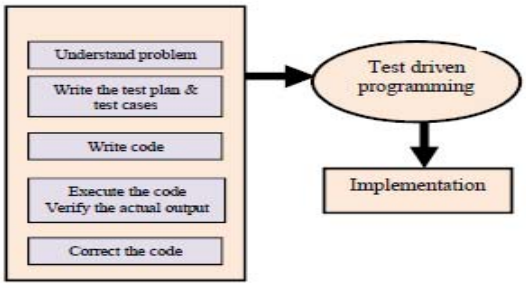

Fig. 2. Test driven approach

Test plan			
Exercise: 1		**Date: 19-9-14**	
Problem Statement: Assume College parking place, contains a single lane that holds up to ten vehicles. There is only a single entrance/exit to the lane. If the student arrives to pick up his vehicle that is not nearest to exit. How can he take out his vehicle? Simulate the scenario.			
Assumptions: Vehicles are arranged in single lane. Maximum 10 Vehicles in lane (Max length=10) Fields considered in the struct are: Vehicle Id (int), Type of vehicle (string), Duration of parking time (float) Data Structure used to implement : Stack			
Constraints: Length of Vehicle Id = 4 digits **Format of Duration of parking time** : hh.mm **Length of Type of vehicle name** = 10 Perform the operations depending on Options: 1-push , 2-pop, 3-display			
Test id	**Input**	**Expected output**	**Actual output**
1	Option=1 Vehicle_Id = 1234 Vehicle_type = 4 wheeler Duration: 0.30	1234 Pushed inside the stack	
2	Option=1 Vehicle_Id = 12344567 Vehicle_type = 4wheeler Duration: 0.30	Invalid	
3	Option=2	Display the top most vehicle information	
4	Option=3	Display all vehicles in the stack	

Fig. 3. Test plan for given scenario

III. RESULTS

Faculty experienced qualitative and quantitative improvement in the process followed by students in writing and testing of the program. Outcome of this approach are:

- Students are able to write quality code starting from first semester itself.
- Better understanding of the problem.
- In Fifth semester time required to complete the implementation is reduced from 5 weeks to 4 weeks.
- Zero time to teach testing at fifth semester miniproject.
- Earlier faculty would spend nearly 2 weeks in teaching testing.
- Use of Test automation using tools is promoted.

IV. CONCLUSION

Spiral learning model is an interactive method of teaching supports learning by making students to build on what is already familiar to them rather than learning new and difficult things. Experience of teachers in using this approach for building software testing skill shows that there is significant increasing the students' learning and the durability of that learning. Quantitatively also the time required for writing quality code is reduced to 50%. This can be recommended for teachers and educators to boost the learning excellence of students as an individual asset for future.

References

[1] Computer Engineering Curricula 2016, Interim Curriculum Report Version: 2015 Oct 25

[2] The guide to the Software Engineering Body of Knowledge SWEBOK® © IEEE – 2004

[3] Fahimeh Veladat, Fatemeh Mohammadi , "Spiral learning teaching method: Stair stepped to promote learning",International Conference on Education and Educational Psychology (ICEEPSY 2011) Available online at www.sciencedirect.com

[4] Mohammed Odeh, "A Reflective Approach to Improve Learning and Teaching of Software Engineering in Large Groups", The International Arab Journal of Information Technology, Vol. 1, No. 0, July 2003

[5] Jalote, P, An Integrated Approach to Software Engineering, 3 ed, Narosa Pub, 2005