Using a Web-Based Testing Tool Repository in Programming Course: An Empirical Study

Anurag Goswami
North Dakota State University
Computer Science Department
Fargo, USA
anurag.goswami@ndsu.edu

Gursimran S. Walia
North Dakota State University
Computer Science Department
Fargo, USA
Gursimran.walia@ndsu.edu

Sameer Abufardeh
North Dakota State University
Computer Science Department
Fargo, USA
Sameer.abufardeh@ndsu.edu

ABSTRACT
This paper highlights an important issue of the knowledge and skill deficiency of software testing among undergraduate students in software engineering discipline. The paper provides an approach for integrating software testing into computer programming course in a non-obtrusive manner. The paper describes the use of the *Web Based Repository of Software Testing Tools* (WReSTT) that can assist the instructors in integrating the testing component into their software engineering course and also provides the students with all the necessary resources (tutorials, quizzes, videos etc) for them to gain general testing knowledge, be able to apply the testing techniques, and become proficient in the usage of testing tools. This paper presents the design of the WReSTT, and then presents an empirical study that was conducted in an introductory computer programming course at North Dakota State University. The results from the study showed that the WReSTT can be used to significantly impact the testing knowledge gained by the students and that the increased use of the WReSTT resulted in a better grade for the students on their programming assignments.

Categories and Subject Descriptors
D.2.5 [**Testing and Debugging**]: *Testing tools (e.g., data generators, coverage testing)*.

General Terms
Measurement, Experimentation, Languages.

Keywords
WReSTT, Software Testing, Unit Testing, Code Coverage, Empirical Study.

1. INTRODUCTION
Software testing continues to be a concept that plagues software engineering students when they enter the software industry because after their undergraduate degrees. On that note, previous researchers have found evidence that the graduating software engineering students and the newly hired software engineering lack basic testing knowledge, testing ability, and the usage of software testing tools [1-4, 8]. This has been identified both by the industrial researchers and the academics as an important knowledge and skill deficiency among the software engineering students. More specifically, the researchers have shown through studies conducted with senior-level undergraduate students that the students do not have the knowledge of different testing techniques, lacked the ability to use testing tool (with code coverage in particular) and created ineffective test cases [1, 2, 3, 4, 8]. While the previous research does not provide or highlight a specific type of testing the student's lack, it is believed that the testing as a whole is an important knowledge deficiency in graduating students and is a focus of attention of this research paper.

While there have been approaches to integrate testing into computer programming courses at various institutions, they do not rigorously enforce the testing in these course especially during the introductory computer programming courses. Furthermore, when testing is used (e.g., upper level courses), there is not enough support for the students to expose them to the testing tools and assist them during the usage of testing tools on their programming assignments. While a lot of testing content is being taught in the upper level courses or the graduate level courses at our institution, there is a lack of focus of integrating testing early in the curriculum. This paper describes the use of the *Web Based Repository of Software Testing Tools* (WReSTT) that can assist the instructors in integrating the testing component into their software engineering course. The WReSTT also provides the students with all the necessary resources (tutorials, quizzes, videos etc) for them to gain general testing knowledge, be able to apply the testing techniques, and become proficient in the usage of testing tools.

The WReSTT also incorporates a collaborative and social networking environment where the students enrolled in a class can communicate with each other, start and contribute to the discussion of software testing related topics, gain and compete for the virtual points (e.g., the number of times each student visited the testing tutorial) for testing related work, and provide greater student involvement, cooperation, and team work through the allocation of the virtual points. The students using the WReSTT can browse different tutorials depending upon the programming language employed in their course.

This paper presents the design of the WReSTT, its main features, and the integration of WReSTT in software engineering courses without affecting the logistics of the course. The paper presents an empirical study that was conducted in an introductory computer programming course at North Dakota State University. The goal of the study was to evaluate the impact of WReSTT on the undergraduate students acquisition of the knowledge of testing objectives, testing techniques, their usage of testing tools, and their proficiency on using the testing tools (in particular the Unit testing and the Code Coverage). The results from the study showed that the WReSTT can be used to significantly impact the testing knowledge gained by the students and that the students are highly likely to no use any other resource or online learning resource if they are not exposed to the WReSTT in their courses. The results also showed that the increased use of the WReSTT

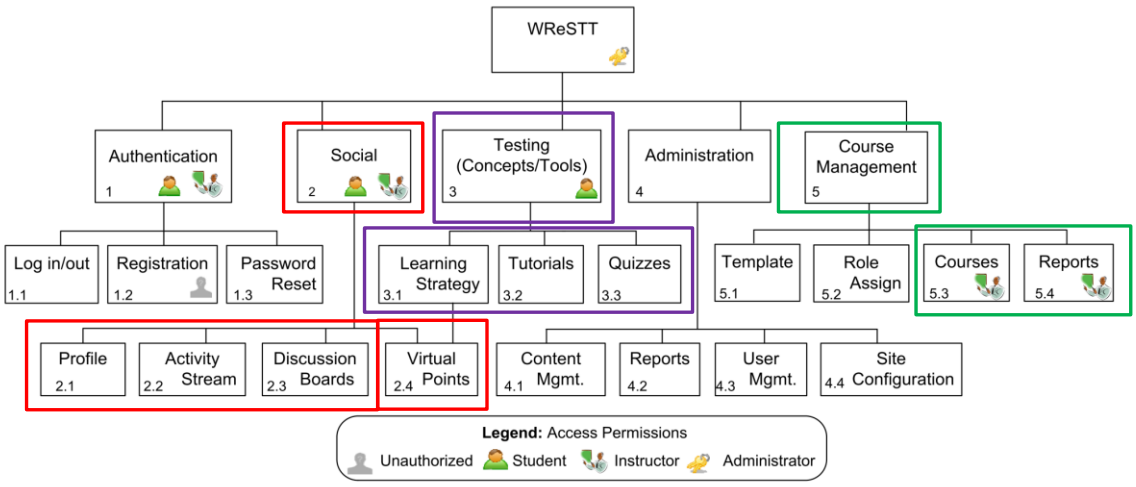

Figure 1. Design of WReSTT [5, 9]

resulted in a better grade for the students on their programming assignments.

The rest of the paper is organized as follows. Section 2 describes the main features of the WReSTT, along with the instructor and the student view of the WReSTT. Section 3 describes the experiment design, the participating students, and the data collected during the experiment run. Section 4 presents the analysis of the data organized around the study goals. The discussion of results is provided in Section 5 followed by the concluding remarks.

2. WEB-BASED REPOSITORY OF SOFTWARE TESTING TOOLS (WReSTT)

This section provides an introduction of the WReSTT and its main features to expose undergraduate students to the testing methods and tools. The following subsection discusses how the WReSTT can be used in programming courses by the instructors (or TA's) and by the students.

2.1 Introduction to WReSTT: Main Features

The WReSTT is designed to be able to provide online learning resources in software testing. The main components of WReSTT are shown in Figure 1 [5, 9] and discussed as: 1) WReSTT provides the students with *reading tutorials* on a variety of software testing concepts and methodologies. Students browse through the testing tutorials, and evaluate their understanding of those concepts through quizzes. 2) WReSTT also provides students with access to the *video tutorials* on different testing tools classified by *category* (Coverage, Metrics, Plug-ins, Test execution, Web), by *language* (C++, Java, VB .Net), and by *test level* (System/UI, Unit). 3) WReSTT provides a *collaborative* and *social networking* learning environment for the students. WReSTT allows students to upload their user profile, post comments on the discussion board, and monitor the activity of other students enrolled in the same class. A unique feature of WReSTT incorporated assignment of "*virtual points*" to the students (e.g., for completing a tutorial and its quiz, posting a testing related comment related to the forum etc) in order to increase student involvement at the individual level as well as at the class level (by allowing students to monitor the virtual point leaders in the same class).

2.2 Using WReSTT: Instructor View

Instructors can use the WReSTT in their courses (that have some programming component) by *creating a course* (e.g., uploading class roster and the assignment of unique ID and password to each student) and through the *course management* (e.g., monitoring the activity streams, generating student reports, and allocation of virtual points for student activities). Figure 2 shows a selection of screen capture (to keep anonymity) that depicts the instructor's interface during the course *creation* and the course *management*.

Course creation: Instructors can use WReSTT at their campus by requesting the access from the administrators (authors) of the WReSTT (link), who will provide the login credentials to the instructor and will create the template for their course title. Next, instructors can login to access their course and use the template instructions to: 1) upload the class roster; 2) create unique login credential for the students; 3) assign students to virtual teams; 4) describe the rubric for the allocation of virtual points for different student activities; 5) enable/disable pre and posttest. There is Help menu available on the WReSTT to assist the course creation.

Course management: After the creation of the course, the instructor can manage the course by: 1) monitoring each student's activity on the WReSTT (e.g., number of times each student visited the tutorial, the time it took them to complete a particular quiz); 2) monitor the virtual points gained by each student in the class; 3) generate and print student reports using different statistics related to the class (e.g., number of students completing a particular quiz, virtual point leaders). Figure 2 shows the instructors' view during the course creation and management.

2.3 Using WReSTT: Student View

As mentioned earlier, students need to authenticate themselves to access the WReSTT for their course. The WReSTT is designed to mirror the social networking tool features to enhance student involvement. Figure 2 (right side) shows the students view after they login to WReSTT.

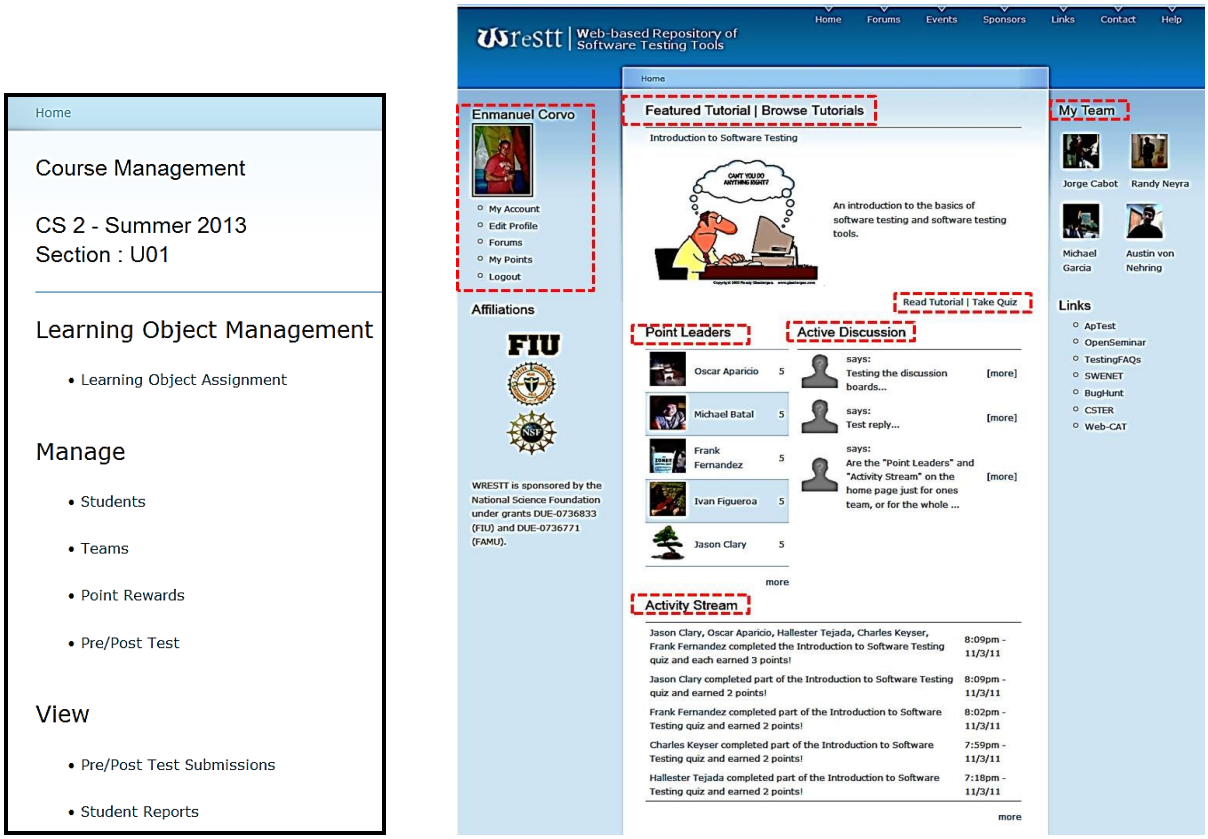

Fig. 2 Instructor view (left side) and the student view (right side)

Some of the main section of students' interface is highlighted in Figure 2. That is, each student can *create profile*, browse the *testing tutorials*, take *quizzes*, watch videos on the *testing tools*, interact with other students in the class via *testing based discussions*, and monitor the *active discussions*, *activity stream* and the *virtual point leaders*. In addition, depending upon the way an instructor wants to run their course, additional virtual bonus may be assigned for the team-based activities (e.g., all members have to complete a quiz to get team virtual points) to foster team collaboration using WReSTT.

3. STUDY DESIGN

The study was designed to investigate the effect of the WReSTT on the undergraduate students' acquisition of general testing concepts, knowledge of testing techniques, and proficiency of testing tool usage in an introductory computer programming course at North Dakota State University. The study utilized a pretest/posttest design in which the participating subjects were pretested on their testing knowledge prior to the introduction of the WReSTT. Next, the students worked individually to complete programming assignments where they were asked to apply the *JUnit* (a unit testing framework for the Java) and the *EclEmma* (code coverage tool) testing tools using the information contained in the WReSTT. After the completion of the assignments, the subjects were tested again (using the same set of questions in the pretest) to evaluate their difference in the knowledge of the testing concepts and testing techniques and testing tools. At the conclusion of the study, the subjects filled a survey to reflect their experience with the use of WReSTT in the current course and their desire to use WReSTT in future programming classes. More details on the study design are presented in the following paragraphs.

Study Goals: This study has two main goals. The first goal is to analyze the impact the WReSTT had on the student's learning of software testing concepts and tools, and is stated formally as:

Analyze student's pretest and posttest scores *for the purpose of* evaluation *with respect to* the impact of WReSTT on the increase in the knowledge acquisition of testing concepts, techniques, and tools *in the context of* undergraduate students enrolled in programming course at a large public university.

The second goal is to analyze the overall satisfaction with the WReSTT features and its usability in a programming course:

Analyze student's post-study questionnaire response *for the purpose of* evaluating *with respect to* the ease of use and integration of WReSTT in programming course.

Participating Subjects: The participating subjects were 21 undergraduate computer science students enrolled in the Computer Science II'2013 course at North Dakota State University. 18 out of 21 students chose to participate in the study.

Artifacts: The students were asked to submit a written report of the *Unit* and the *Code Coverage* testing tools and techniques used, the test cases, the test execution results during the testing of their code, and their understanding of the results achieved. The artifacts were evaluated by the instructor of the course (not involved during the design of the empirical study).

Study Procedure: The study steps are as follows:

a) *Step 1 – Pretest:* A pretest was administered at the beginning of the study. The test instrument is shown in Appendix A. The goal of the pretest was to measure the student's knowledge of testing concepts and tools prior to using the WReSTT. The questions on the pretest measured the baseline knowledge of the testing concepts, knowledge of testing techniques and testing tools, and the proficiency in testing tools usage.

b) *Step 2 – Training on WReSTT*: Next, the subjects were trained on how to access the WReSTT for browsing tutorials, taking quizzes, posting and viewing discussion threads. Also, the students were taught how to use the WReSTT for watching testing tutorials on different testing techniques and testing tools.

c) *Step 3 – Programming Assignment Description*: Using the information in the WReSTT, the students worked individually on their programming assignments and test their code at the Unit level (using JUnit tool to test classes), and to evaluate the test coverage achieved on their programs (using EclEmma tool). The students used the tutorials on the JUnit and the EclEmma tools to learn how to apply these tools. The students then documented the report of the test execution results along with the source code which was evaluated by the course instructor (not a part of the research team).

d) *Step 5 – Posttest*: At the end of the course assignments, the students were re-tested on their knowledge of testing techniques and tools using the same instruments used during the Step 1 (pretest). The goal of the posttest was to measure the impact of WReSTT on the increase in testing knowledge.

e) *Step 6 – Post-Study Survey:* Finally, the subjects were asked to fill a survey to evaluate the usability and the usefulness of the WReSTT in introductory programming course. The survey questions (shown in Appendix B) included questions related to the student's overall reaction to the WReSTT, questions related to the usefulness and adequacy of the testing tutorials and tools in WReSTT, and open ended questions to gather feedback on improving the usage of WReSTT in future courses .

Data Collection and Evaluation Criterion: The data includes the student's responses on the pretest and the posttest survey. The responses (on pretest and posttest) were evaluated by assigning a value representative of the adequacy of their response. More details are provided in subsection 5.1.

The student's responded to the survey questions using the 5-point likert scale: 1 = Strongly Disagree, 2 = Disagree, 3 = Neither agree nor Disagree, 4 = Agree, 5 = Strongly Agree. We treated these scales as interval scales (rather than the ordinal scales), following the standard practice in the social sciences [ ]. The survey responses for each question were averaged across all the students and evaluated for statistical significance (i.e., whether the average is significantly greater than the middle point of the scale). The subject's responses to open ended questions were collected to help researchers better understand the results. More details are provided in subsection 4.2

We also collected the grade received by students on the testing assignments to correlate their perceived usefulness of WReSTT on their grades.

4. RESULTS AND ANALYSIS

The results are organized around study goals presented in Section 3. Section 4.1 compares the results regarding the students testing knowledge after and before using the WReSTT. Section 4.2 evaluates the students' feedback on the use of WReSTT in programming assignments. In addition, this section presents the correlation between the usefulness of WReSTT against the performance on the assignments.

4.1 Pretest vs. Posttest Results

To evaluate the impact of the WReSTT on the students' knowledge acquisition of the testing concepts and their proficiency of the testing tools and techniques, a comparison of the pre-test and post-test results was performed. The students were assigned a score representative of their responses to the questions on the pretest and posttest. The rubric for the expected answers for each question is shown in Appendix A that was used to calculate the total score for each student. Next, the average score of all the eighteen students on the pretest vs. the average score on the posttest were compared. The result showed that the subjects, performed better on the posttest (an average score of 6.43) as compared to their performance on the pretest (an average score of 13.85). A one-sample ANOVA test showed that the increase in the overall testing knowledge (measured by the increase in the average score of subjects during the posttest when compared to their. Pretest score) after being exposed to the WReSTT is statistically significant ($p < 0.05$).

While this result in interesting, we wanted to evaluate the usefulness of WReSTT seperately on the students' increase in the a) general software testing knowledge, b) testing tool usage, and c) the proficiency level of the testing tools usage.

General testing knowledge: The student responses to the questions 1, 2a, and 2b were evaluated during the pretest and during the posttest. As mentioned earlier, the students were assigned scores for each question based on the correctness of their response (see Figure 3 for the questions and the rubric). Then, the scores on these questions were added to indicate the student's general knowledge of software testing and the testing technique.

1. What is the main objective of program testing?
To find bugs/errors/faults in a program or any sentence that conveys this idea.

2.(a) Identify all the testing techniques you have used to create test cases:
Equivalence partitioning, boundary analysis, random selection, state-based, among others.

(b) Write test cases to test the code provided below. Identify the testing technique used to generate the test case.

```
//Method to withdraw money from a bank account
//The following variables are defined as follows: requiredMin = 50.0 and balance = 100;
public void withdraw (double amount) {
  if ((balance - amount) < requiredMin)
    System.out.println("Insufficient funds");
  else
    balance = balance - amount;
}
```

Input Value	Expected Output Values			Testing Technique
amount	amount	balance	requiredMin	

(b)(i) Numeric values for all the attributes in the table were given.
(ii) Values were checked for correctness based on the semantics of the method.
(iii) Any of the answers in 2(a) in conjunction with branch coverage, statement coverage, etc..
(iv) The input value provided a guide to whether or not a particular testing technique was used, e.g., amount = 49.9 would signify boundary analysis, amount = -1 equivalence partitioning

Fig. 3 General testing knowledge question and expected answers

The average scores of the "*general testing knowledge*" of students increased from an average of 0.45 during the pretest to an average score of 2.12 during the posttest. A one-sample ANOVA test demonstrated a significant increase in the general testing knowledge of students ($p<0.05$) and a significant increase in the number of students that were able to apply at least one particular testing technique (based on the question 2 (b) responses) during the posttest. Therefore, the WReSTT was able to improve the students understanding of general testing concepts and their ability to apply a particular technique to test the code.

Testing tool usage: The questions 3 and 4 (as shown in Figure 4) were used to evaluate the students testing tool knowledge and its usage. The scores for questions 3 and 4 were added to indicate their usage of testing tools. The results showed that, on average, the subjects knowledge and the usage of testing tools was significantly higher during the posttest (an average of 0.58) when compared to their pretest score (an average of 3.2) at $p<0.01$ level. Furthermore, the students during the posttest were able to identify a significantly larger number of tools for each category (i.e., Unit testing, Functional testing, Code Coverage) when compared to their responses during the pretest.

The students also rated their proficiency levels in using the testing tools (part of question 4(b)). This was done to understand whether the perception of their proficiency for specific tools also increased (in addition to their actual ability). The students rated their proficiency for Unit testing, functional testing, and code coverage on a scale of 1-5 with 1-barely competent and 5-extremely competent. The results showed that, on average, the subjects felt

3. Have you ever used tools to support testing of programs? Circle your answer:
   Yes No
4. If you have answered "Yes" to Question (3), answer the following questions:

   a. List the names of the tools you have used:
   JUnit, PhPUnit, OCUnit, GHUnit, Visual Studio Unit Test, IBM Functional Tester, SWAT, Cobertura, EclEmma, Cover Story

   b. List one tool in each of the following categories that you have used and indicate your level of proficiency corresponding to each tool on a scale of 1 - 5 with, *1 = barely competent* and *5 = extremely proficient*:

	Category	*Tool(s)*	*Proficiency*
i	Unit Testing	***JUnit, PhPUnit, OCUnit, GHUnit, Visual Studio Unit Test***	
ii	Functional Testing	***IBM Functional Tester, SWAT***	
iii	Code Coverage	***Cobertura, EclEmma, Cover Story***	

Fig. 4 Testing tool usage questions

more proficient in the testing tool usage (when averaged across all the three categories for all the students) during the posttest as compared to their proficiency of tools during the pretest. A Paired-sample Wilcoxon Signed-Rank test to compare the average of the ratings at pretest vs. posttest found the increase in the perceived proficiency to be statistically significant ($p = 0.012$).

Next, questions 5, 6, and 7 were used to gather evidence regarding other online resources that the students used for learn testing and the type of information available on those resources. The students generally did not report any other online source. Regarding question 7, the results from a One-sample Wilcoxon Signed-Rank test showed that the average ratings of the "*benefit of using tools to support testing of programming assignments*" after using the WReSTT (at posttest) was significantly greater than 3 (i.e., midpoint of the scale) at $p<0.05$. Furthermore, a Paired-sample Wilcoxon Signed-Rank test revealed significant increase in the students' perception of benefits during the posttest when compared to their rating at the pretest ($p < 0.01$).

4.2 WReSTT Survey Results

After completing the posttest, the students were asked to complete a post-study survey that evaluated the student's overall response to the WReSTT, followed by more specific evaluation of the

5. Do you know of any online resources that provide information on software testing?
   Yes No
6. If you have answered "Yes" to Question (5), answer the following questions:

   a. State the names of online resources.
   WReSTT, Apple Developer, Wikipedia, YouTube, Udacity, among others

   b. State the type of materials (notes, lab exercises, tutorials etc.) found at each resource listed above.
   Tutorials, Video Tutorials, Tool Documentation, Quizzes, Notes on Testing, among others

7. How beneficial do you think it is to use tools to support testing of your programming assignments? Use a scale of 1 - 5 with *1 = Not beneficial* and *5 = Extremely beneficial*

Fig. 5 Rest of the questions

testing tutorials in WReSTT. The results of the students ratings (using a 5-point likert scale ranging from 1-Strongly Disagree to 5-Strongly Agree) is discussed in this subsection.

The first question in the survey collected a response (Yes/No) to determine what percentage of the students have ever used a learning source other than WReSTT to learn about testing concepts and tools. Based on the response from eighteen students, 83% (fifteen) of the students indicated no use of any other resource. This shows a lack of exposure of software testing in introductory programming course (in particular at our institution).

Regarding the feedback on the usability of the WReSTT, students rated the WReSTT on items 2 through 15 using a 5-point scale on different attributes (e.g., ease of use, ease of learn, expected functionality, clarity of information, recommendation) as shown in Table 1. The first two columns of Table 1 show the number and the description of the attribute being evaluated. The third column of Table 1 reports the mean and the standard deviation (S.D.) score from all the eighteen subjects for each attribute.

To evaluate the each attribute of WReSTT, we conducted a One-sample Wilcoxon Signed-Rank test to determine whether the mean ratings were significantly greater than 3 (i.e., midpoint of the scale). The test indicates that the WReSTT received significantly positive ratings on all fourteen attributes (i.e., $p<0.05$). The last column of Table 1 shows the p-value for each attribute.

Next, we evaluated the student's perception of the usefulness of the testing tutorials in WReSTT. For this purpose, students rated items 16 through 21 (as shown in Table 2) using a 5-point scale. Items 16-20 measured the student's response on usefulness of WReSTT tutorials in terms of overall quality, quantity, and with respect to its use on how to use the unit testing / code coverage / functional testing tools. The average score (S.D.) of each item is shown in Table 2. The result from a One-sample Wilcoxon Signed-Rank test that evaluated whether the mean ratings were significantly greater than 3 (i.e., midpoint of the scale) is shown in the last column of Table 2. The results showed that the WReSTT tutorials received significantly positive ratings of all the characteristics ($p <0.05$).

Table 1. Overall response to the WReSTT

#	Attribute	Mean (S.D.)	p-value
2	Overall, I am satisfied with how easy it is to use the website	3.92 (0.61)	<0.01
3	It is simple to use the website	3.78 (0.69)	<0.01
4	I feel comfortable using the website	3.5 (0.65)	<0.01
5	It was easy to learn to use the website	3.78 (0.57)	<0.01
6	I believe I became productive quickly using the website	3.35 (0.92)	<0.05
7	The information (such as online help, on page messages, and other documentation) provided with the web site is clear	4.07 (0.75)	<0.01
8	It is easy to find the information I need	3.78 (0.69)	<0.01
9	The information is effective in helping me complete the tasks and scenarios	3.61 (0.5)	<0.01
10	The interface of the website is pleasant	3.85 (0.77)	<0.01
11	I like using the interface of this website	3.64 (0.49)	<0.01
12	The website has all the functions and capabilities I expect it to have	3.78 (0.84)	<0.01
13	I believe that the website helped me earn a better grade	3.3 (0.65)	<0.05
14	I would recommend the website to fellow students	3.71 (0.82)	<0.01
15	Overall I am satisfied with the website	3.84 (0.64)	<0.01

Regarding the feedback collected from open ended questions, a significantly large number of students would like to introduce WReSTT in their future programming courses.

5. DISCUSSION OF RESULTS

The results from this study indicate the promise of using WReSTT to teach software testing in introductory computer programming courses. The students understanding of testing concepts, their knowledge of testing techniques, their tool usage and their perceived proficiency of the tool usage showed a significant improvement due to the exposure to the WReSTT. The results also show that the students had a significantly positive feedback of the WReSTT in terms of its usefulness, its ease and clarity of the information. Also, the results from the post-study survey showed that the WReSTT tutorials helped them understand the concepts and tools better and that they would not have used testing tools if they were not exposed to the WReSTT. An interesting result was the student's perception that the WReSTT helped them achieve a better grade in this class. We wanted to evaluate if this perception was true in reality by analyzing whether there was any positive correlation between the actual points scores on the programming assignments by the students (where WReSTT was used) vs. their rating of the belief that the website helped me earn a better grade. The result from a linear regression showed a significantly positive ($p= 0.028$, $r^2= 0.42$), that the students who had a significantly positive perception of WReSTT on a better grade achieved higher points on their programming assignments (where WReSTT was used).

6. CONCLUDING REMARKS

Based on these results, we would like to continue investigating the further use of WReSTT in other software engineering courses at NDSU and other institutions. We would also welcome collaborations with the researchers at other institutions to help generalize the study results. Since this was a first study of using WReSTT at NDSU, we did not used the "virtual points" concepts and some other social networking features (See Figure 1) in this study. We have already begin using these features in studies that are undergoing in the Fall 2013 semester at our institution. The results from these studies would help us provide better understanding of the impact of WReSTT's collaborative learning environment on the students learning of software testing concepts and tools over a course of the semester.

7. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grant DUE-1225972. We would like to thank the students who participated in the study and the reviewers of the

Table 2. Response to the usefulness of testing tutorials in WReSTT

#	Attribute	Mean (S.D.)	p-value
16	The tutorials in WReSTT helped me to better understand testing concepts	3.84 (0.68)	<0.01
17	The tutorials in WReSTT helped me to better understand how to use unit testing tools	3.69 (0.75)	<0.01
18	The tutorials in WReSTT helped me to better understand how to use code coverage testing tools	3.41 (0.66)	<0.01
19	The tutorials in WReSTT helped me to better understand how to use functional testing tools	3.36 (0.8)	<0.05
20	The number of tutorials in WReSTT is adequate	3.61 (0.76)	<0.01
21	I would not have used testing tools in my project if WReSTT did not exist.	3.92 (0.75)	<0.001

study design and the earlier version of the paper

8. REFERENCES

[1] Timothy C. Lethbridge. 2000a. Priorities for the education and training of software engineers. J. Syst. Softw 53, 1 (July 2000), 53–71. http://dx.doi.org/10.1016/S0164-1212

[2] Andrew Begel and Beth Simon. 2008a. Novice software developers, all over again. In Proceedings of the Fourth international Workshop on Computing Education Research (ICER '08). ACM, New York, NY, USA, 3–14. http://dx.doi.org/10.1145/1404520.1404522 .

[3] Jeffrey C. Carver and Nicholas A. Kraft. 2011. Evaluating the testing ability of senior-level computer science students. In Software Engineering Education and Training (CSEE T), 2011 24th IEEE-CS Conference on. 169–178. http://dx.doi.org/10.1109/CSEET.2011.5876084

[4] Vahid Garousi and Tan Varma. 2010. A replicated survey of software testing practices in the Canadian province of Alberta: What has changed from 2004 to 2009? J. Syst. Softw. 83, 11 (Nov. 2010), 2251–2262. http://dx.doi.org/10.1016/j.jss.2010.07.012

[5] WReSTT Team. Web-based Repository for Software Testing Tools, 2010. http://wrestt.cis.fiu.edu/ .

[6] ACM/IEEE-CS INTERIM REVIEW TASK FORCE. 2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001. http://www.acm.org/education/curricula/ComputerScience2008.pdf

[7] ASTIGARRAGA, T., DOW, E., LARA, C., PREWITT, R., AND WARD, M. 2010. The emerging role of software testing in curricula. In Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments, 2010 IEEE. IEEE, Piscataway, NJ, USA, 1–26

[8] T. C. Lethbridge, J. Diaz-Herrera, R. J. J. LeBlanc, and J. B. Thompson. Improving software practice through education: Challenges and future trends. In FOSE '07: 2007 Future of Software Engineering, pages 12–28, Washington, DC, USA, 2007. IEEE Computer Society.

[9] P. J. Clarke, A. A. Allen, T. M. King, E. L. Jones, and P. Natesan. Using a web-based repository to integrate testing tools into programming courses. In Proceedings of the ACM OOPSLA 2010 Companion, SPLASH '10, pages 193–200, New York, NY, USA, 2010. ACM.

[10] P. J. Clarke, J. Pava, Y. Wu, and T. M. King. Collaborative web-based learning of testing tools in se courses. In Proceedings of the 42nd SIGCSE Conference, pages 147–152, New York, NY, USA, 2011. ACM

Appendix A

PRETEST/POSTTEST

This test will NOT impact your course grade.

ID Number (DO NOT use your name):

1. What is the main objective of program testing?

2.(a) Identify all the testing techniques you have used to create test cases:

(b) Write test cases to test the code provided below. Identify the testing technique used to generate the test case.

```
//Method to withdraw money from a bank account
//The following variables are defined as follows:
//requiredMin = 50.0 and balance = 100;
 public void withdraw (double amount) {
  if ((balance - amount) < requiredMin)
   System.out.println("Insufficient funds");
  else
   balance = balance - amount;}
```

Input Value	Expected Output Values			Testing Technique
amount	amount	balance	requiredMin	

3. Have you ever used tools to support testing of programs? Circle your answer: Yes No

4. If you have answered "Yes" to Question (3), answer the following questions:

a. List the names of the tools you have used:

b. List one tool in each of the following categories that you have used and indicate your level of proficiency corresponding to each tool on a scale of 1 - 5 with, *1 = barely competent* and *5 = extremely proficient*:

	Category	*Tool(s)*	*Proficiency*
i	Unit Testing		
ii	Functional Testing		
iii	Code Coverage		

5. Do you know of any online resources that provide information on software testing? Yes No

6. If you have answered "Yes" to Question (5), answer the following questions:

a. State the names of online resources.

b. State the type of materials (notes, lab exercises, tutorials etc.) found at each resource listed above.

7. How beneficial do you think it is to use tools to support testing of your programming assignments? Use a scale of 1 - 5 with *1 = Not beneficial* and *5 = Extremely beneficial*

8. If you answered 2 or above to Question (7), state one reason for your answer.

.