AC 2009-433: A STATE-OF-PRACTICE ON TEACHING SOFTWARE VERIFICATION AND VALIDATION

Zulfa Zakaria, IIUM

Zulfa Zakaria is a lecturer with the Kulliyyah of Information and Communication Technology (KICT)at the International Islamic University Malaysia (IIUM)since November 2002. She was undergraduate Information Technology student at the Northern University Malaysia from 1992 until 1996. Zulfa has a Master in Real Time Software Engineering from the Technology University Malaysia. From 2006 to 2007, she has been a researcher at the University of Queensland. She is currently pursuing her PhD in software engineering at the Putra University Malaysia.

A State-of-Practice on Teaching Software Verification and Validation

Abstract

Software testing is an essential activity in the software development process. Moreover, it covers a large part of the development costs involved. And, as the software tester is a key player in the testing activities that occur throughout the software lifecycle, the efficacy of such testing depends very much on his or her experience, efficiency, skills, and intuition. Given this importance of software testing, educators face a significant challenge when teaching and equipping students with the testing methodologies, skills, and knowledge that are in line with industry needs. The aim of this paper is to examine the current state-of-practice with respect to teaching Software Verification and Validation (V&V) in the accredited Bachelor of Software Engineering (BSE) programs offered in Australian universities. An online survey was conducted to obtain the required data from these universities. The same online survey was used to obtain data from the Certified Software Test Professional (CSTP) program. Nine universities responded and the data obtained are compared to the data relating to the CSTP program. Consequently, this paper reports on the differences between the two programs by focusing on the methods of teaching and the coverage of those topics specified within the area of Software V&V Knowledge as it is stated in the ACM Curriculum Guidelines. The survey's results and analysis facilitate the discussion of further details concerning what is lacking in the teaching of each topic for each category. An initial suggestion that would contribute to overcoming this educational shortfall is explained briefly in the Future Work section.

1. Introduction

Software testing comprises a process of revealing any errors that may exist within a particular piece of software [1]. The Verification and Validation (V&V) procedures applied involve both static and dynamic techniques of system checking designed to ensure that the tested program satisfies its specifications and meets the expectations of the stakeholders [2]. It is the software tester who is responsible for constructing and executing such tests. As a consequence, the testing methodologies, skills, and knowledge of the tester are important qualities because their effective application will establish confidence among end users that the software is reliable.

Despite the importance of such testing within the software industry, it receives little attention in the undergraduate curriculum [3], with the coverage in Computer Science (CS) and Software Engineering (SE) courses assessed as insufficient [4]. Technology is rapidly changing and this implies that instructors must instill in CS and SE students the testing skills, methodologies, and knowledge required to meet the challenges of this dynamic industry. Consequently, the institutions concerned should regularly review their courses and programs and make any changes necessary to ensure that curricula do not lag behind industry needs [2]. In response to this concern, the first objective of this paper is to examine the current state-of-practice in the teaching of Software V&V within the BSE programs offered in Australian universities that are accredited by Engineers Australia and the Australian Computer Society. The examination is based on the Software V&V Knowledge requirements set out in the Association for Computing Machinery (ACM) Curriculum Guidelines [2]. The Guidelines result from the joint curriculum task force comprising representatives from the ACM and the Computer Society of the Institute for Electrical and Electronic Engineers [5]. These are discussed further in the Survey Method section below.

The other matter of importance and concern that emerges in this context is the very real shortage of software testing professionals who possess the required testing skills, methodologies, and knowledge within the Australian industry. Ng et al. [6] reported that in this industry, seven organizations (15.9% of the total) had 60% to 79% of the members of their testing team who were formally trained. At the same time, there were 10 organizations (22.7%) that had among their testers over 80% who had completed a formal software testing education at a university and 15 organizations (34.1%) with less than 20% of their testers in this category [6]. For this reason, the second objective of this paper is to compare the university BSE and the Certified Software Test Professional (CSTP) programs by focusing on the methods of teaching and the coverage of topics comprising Software V&V Knowledge. The CSTP program has been chosen for comparison because it provides instruction in the relevant testing skills, methodologies, and knowledge within its fundamental and advanced courses [7]. The CSTP program is one that specifically recognizes those interested in being software testing specialists and as the survey results and analysis here reveal, differences do exist between this program and the available BSE programs.

A survey is the optimum method to apply when investigating problems in this particular situation. For the present purpose, an online survey was implemented during a five-month period from the middle of February until June 2007. It covered 16 Australian universities that offered accredited BSE programs in Australian universities and was undertaken after being successfully subjected to the required ethical review process (see the form in the Appendix below). Obtaining the relevant information from these universities began with emails being sent to the respective BSE coordinators requesting details of their Software V&V courses and the lecturers involved. A follow-up phone call was made if a coordinator did not reply. Once such information was obtained, an email was sent to all the lecturers nominated to invite them to participate in the online survey concerning the teaching of Software V&V. A gentle reminder was sent to any lecturer who did not respond initially. All the information obtained was compiled, analyzed, and written up in a report. Often, such surveys in the field of software engineering suffer from low response rates because it is difficult to obtain feedback from respondents [8]. Here, the data obtained came from only nine of the Australian universities. The same process of obtaining information was applied to the course instructors in the CSTP program offered by K. J. Ross & Associates (KJRA) and Object Training, the commercial training division of Object Consulting Pty Ltd. These organizations provide comprehensive training in accordance with the accredited CSTP program [7].

For educators, teaching and equipping BSE students with the necessary testing methodologies, skills, and knowledge that are in line with industry needs represent very challenging tasks. Another important factor of serious concern is the effectiveness of incorporating software testing into already over-packed curricula [9]. Software testing is relevant in BSE curricula because it is likely to be an important part of the professional life of most graduates. The survey results obtained in the present context show that there is a significant lack of coverage in the teaching of Software V&V topics within BSE programs. An initial suggestion for overcoming this shortfall is detailed and explained in the Future Work section below.

The next section, Section 2, discusses related work in the field of software testing. Section 3 outlines the survey method and Section 4 presents the survey results and analysis, including details of possible challenges to their validity. Section 5 explains the possibilities for future work, while Section 6 draws some conclusions. Section 7 expresses thanks to the respondents and to K. J. Ross & Associates for their assistance.

2. Related Work

Several extant research studies have investigated the teaching of software testing within undergraduate and postgraduate CS or SE curricula with particular reference to the challenges involved. The discussion in this section groups these studies according to four strategies that relate to the design of software testing courses, empirical studies examining methods of teaching software testing course, the SPRAE software testing framework, and the implementation of testing tools for teaching software testing.

The first such strategy was involved in six research studies that focused on introducing software testing courses at the postgraduate and undergraduate level. Several researchers have suggested using a mixed lecture and practical-based program [10], [11], [12], [13], [14], [15]. Kaner [13] introduced simple examples and verbal instances of industry practices when teaching domain testing, while Padmanabhan [14] reported on teaching the same course by means of instruction effectively combining explanatory and procedural material. Both of these authors agreed on the importance of developing cognitive strategies for teaching this coursework so as to ensure that students develop higher-order thinking about the domain testing task.

In the case of the second strategy, a series of empirical studies have examined methods of teaching for Test-driven Development (TDD), agile practices, and testing an application through an API [15]. This study involved the researchers implementing lecture-based courses with additional methods included, such as take-home assignments, open-book examinations, and the creation of a TDD program from scratch. The researchers found that the students lacked the capacity for higher-order thinking once they had to apply the TDD concept under examination conditions.

The third strategy emphasized the combination of 80% practice and 20% theory of testing within programming courses by using the SPRAE software testing framework and varieties of testing activities that included, inter alia, grading another students' programming, treasure hunting, and writing test cases before writing programs [16]. The SPRAE activities encouraged students to work in teams and to communicate with each other in solving problems. Access to the SPRAE framework enables senior students who have completed programming courses to assist the lecturers as tutors to junior students. This strategy encourages the students to practice and apply testing concepts throughout their degree studies.

The fourth strategy that is pertinent here emphasizes that it is essential to use testing tools for teaching software testing [17], [18], [12], [19], [15]. This is based on the JUnit-style except for the Multi-tool and PGMEN-tool cases. The benefits of these testing tools are set out below.

- Highly Extendable and Flexible (HF2) tools allow students to experience the difficulties associated with unit testing [17].
- The BlueJ tool is capable of recording simple object creation and interacting sequences as JUnit-style test cases, incorporating object interaction with the JUnit to perform regression testing, helping students to write tests from the beginning, and matching them with an objects-first pedagogy [18].
- The DrJava-tool [20] provides built-in support to help students write the JUnit-style test cases for their classes as required [18].
- The Multi-tool focuses on maintenance operations [15].
- The PGMEN-tool generates batch test drivers and assists students in writing their own test cases and test suites [12].

Each strategy has its own strengths when it comes to improving and incorporating software testing into CS or SE curricula. Nevertheless, there is a need to examine further the methods of teaching and learning that can enhance the outcomes for BSE students with respect to analytical thinking and elicit the testing tools and activities that can equip them with the testing skills required to meet industry needs. These needs include defining test objectives, designing test cases, documenting test results, reusing the same test cases after changes are made to the software, and redesigning test cases based on the analysis of previous test results [6]. To establish these industry needs, the Pex; white-box testing tool can be further examined as a means of supporting teaching SE students in Software V&V courses [21]. The link to the Pex website is http://research.microsoft.com/en-us/projects/pex/ and its capabilities are set out below.

- Produces traditional unit tests cases with high code coverage.
- Suggests a bug fix once a generated test fails.
- Records detailed execution traces of test cases.
- Performs a systematic program analysis.
- Encourages TDD practice.

3. Survey Method

3.1. ACM Curriculum Guidelines

The ACM Curriculum Guidelines are intended to assist in SE curriculum development. The survey questions devised depended mainly on the ACM documentation and focused on Software V&V Knowledge. Five categories were involved [2]:

- Category 1 (V&V Terminology and Foundations)
- Category 2 (Reviews)
- Category 3 (Testing)

- Category 4 (Human Computer User Interface Testing and Evaluation)
- Category 5 (Problem Analysis and Reporting)

An additional category included in this survey was designated Category 6 (Development Paradigms). However, it is not discussed here because it is minimally related to Software V&V Knowledge. A sample of the online questionnaire included in the Appendix below.

3.2. Survey Objectives

Every survey must begin with a consideration of its objectives so as to ensure that the outcomes intended are effectively achieved. Here, the pertinent survey objectives were as follows.

- To examine the amount of coverage of each topic in the five knowledge categories within the BSE programs.

- To examine the methods of teaching used in each category.

An additional objective was to examine the current practices of teaching Software V&V in the CSTP program.

3.3. Survey Descriptions

At the beginning of the survey, the participants were asked to fill in their course name and course code. They were not required to provide any personal details. A glossary page was provided to assist participants with understanding the meaning of each topic and a general comments section was included to allow them to provide any further relevant information. The online survey comprised two main parts: Part One: Amount of coverage, and Part Two: Methods of teaching. The details of these parts are summarized below.

3.3.1. Part One: Amount of Coverage

The data collected for Part One comprised the total number of relevant courses, including both required and elective courses. This part was concerned with the first objective of the survey. Additionally, the collection of courses is summarized within one BSE program per university. The extent of coverage assessment consisted of five possible answers: None (N), Just a Mention (JM), Teach the Basics (TB: 1–2 hours), Teach Moderately (TM: 2–3 hours), and Teach in Depth (TD: more than 3 hours).

3.3.2. Part Two: Methods of Teaching

Six methods of teaching were considered: lectures, tutorials, practicals, assignments, projects, and examinations. A lecture was identified as a theoretical type of teaching while the rest were designated as based on practice. Part Two was concerned with the second objective of the survey.

3.4. Survey Method

Information about the BSE programs and their Coordinators for the respective universities was obtained through searches of the Faculty and School web sites. The survey participants were lecturers or instructors from nine universities who had Software V&V knowledge and experience in teaching in the BSE program. In general, the participants' backgrounds enabled them to meet the requirements and objectives of the survey, especially with respect to answering questions relating to the state-of-practice in teaching Software V&V. Subsequently, the lecturers identified were approached by means of an email. The email indicated to them that all survey data, comments, and responses would be treated as strictly confidential and all collected material would be accessible only by the research team and used solely for statistical purposes and to provide directions for future research. The aggregated results would remain anonymous and there would be no identification of individual lecturers or programs.

3.5. Sample Selection

The survey was cross-sectional, with participants asked for information relating to a fixed point in time [22]. It used systematic sampling that involved every respondent from the population list [23]. Accredited BSE programs in 16 Australian universities were initially targeted and the survey focused on participants who taught courses related to Software V&V during 2006. A specific year was chosen to guarantee that data collection avoided mixing up the curricula for different years. This restriction was intended to ensure that the universities had made no changes to any of the courses that would skew the results. All Australian universities adhere to the same accredited degree name, Bachelor of Engineering (Software Engineering). To mitigate cost and time factors, an online survey was used. Also, in response to necessary ethical considerations, the survey was completely voluntary and participants were free to withdraw at any point.

4. Survey Analysis and Results

The complete survey results obtained from nine out of the total number of universities approached related to required and elective BSE courses encompassing Software V&V. Only two universities, A and B, offered both such required and elective courses, while six universities, C, D, E, F, G, and H, offered required courses only. The last of the universities, I, only offered one elective course.

The numbers of pertinent courses offered were identified as follows.

- University A (three required courses and two elective courses)
- University B (two required courses and one elective course)
- University C (two required courses)
- University D (two required courses)
- University E (three required courses)
- University F (five required courses)
- University G (three required courses)
- University H (four required courses)
- University I (one elective course)

The comparable CSTP program consists of one fundamental course and three advanced courses. Its Foundation Course embraces three modules: Module 1 (Test Fundamentals, Life Cycles, and Strategies); Module 2 (Test Design Techniques), and Module 3 (Testware, Test Process, and Documentation). An Advanced Course embraces eight modules in three streams: the non-functional testing stream (three modules), the test automation stream (three modules), and the test management stream (two modules).

4.1. Survey Data and Analysis of Results

The data from each university listed required and elective courses. It was difficult to compare the respondent universities fully by means of examining all the courses offered. The process became tedious because University A, for example, had five courses to consider, University B had three courses to consider, and so on with the others. Therefore, to establish the amount of coverage for each topic in each category for all the courses, composite ratings were used as set out in Table 1.

Horizontal View

Table 1. Composite ratings

Vertical View

+	N	JM	TB	TM	TD
N	N	JM	TB	TM	TD
JM	JM	JM	TB	TM	TD
TB	TB	TB	TM	TD	TD
TM	TM	TM	TD	TD	TD
TD	TD	TD	TD	TD	TD

For instance, the calculation of the coverage for **Topic A1 (Objectives and Constraints of V&V)** in the case of University A is shown in Table 2 (see the results for University A in the Appendix below).

Table 2. Calculation performed based on composite ratings

A1 = Course 1 + Course 2 + Course 3 + Course 4 + Course 5
A1 = TB + JM + JM + N + TM			

Formula	**Vertical View**	**Horizontal View**	**Result**
TB + JM + JM + N + TM	(take bold character) TB	JM	TB
TB + JM + N + TM	(take bold character) TB	JM	TB
TB + N + TM	(take bold character) TB	N	TB
TB + TM	(take bold character) TB	TM	TD
A1 = TD (answer for topic A1 which is stated on the last column in Appendix)			

4.1.1. Part One: Amount of Coverage

The following results are based on comparing the university BSE programs with the CSTP program. All relevant required and elective courses within all the respondent universities were analyzed. If the comparison is based on required courses and the CSTP Foundation course only, there is no significant difference in the coverage of Software V&V topics. Adding both required and elective courses for each university to gain a reasonable coverage and then comparing the courses with the fundamental and advanced courses of the CSTP program revealed relatively significant gaps in the coverage of the topics within BSE programs. The results of this analysis provide some insights that will prove useful when considering the future direction of teaching software testing.

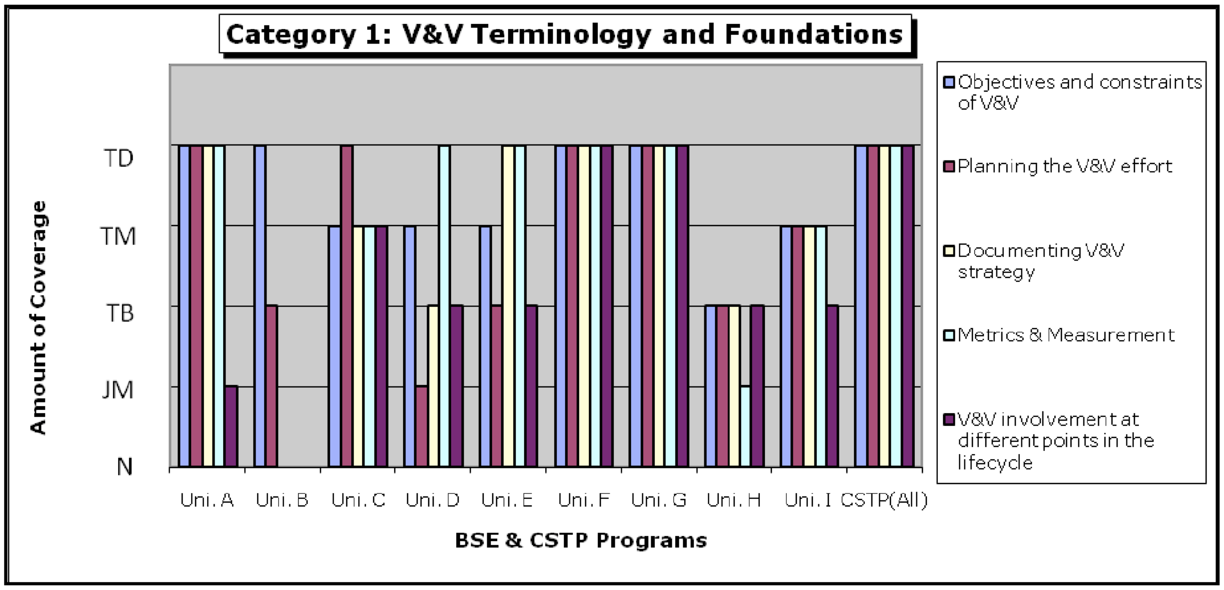

Figure 1. Category 1: V&V terminology and foundations.

Category 1 consists of five topics relating to software testing: the objectives and constraints of V&V, planning the V&V effort, documenting the V&V strategy, metrics and measurement, and V&V involvement at different points in the software lifecycle. In the ACM Curriculum Guidelines [2], five hours are specified for teaching this category. The CSTP program and two universities (F and G) covered all these topics in depth. Significant gaps existed in University B which taught only two topics, objectives and constraints of V&V and planning the V&V effort. Most topics as covered in the other universities extended to more than the basics, while the topic with the least emphasis was V&V involvement at different points in the software lifecycle.

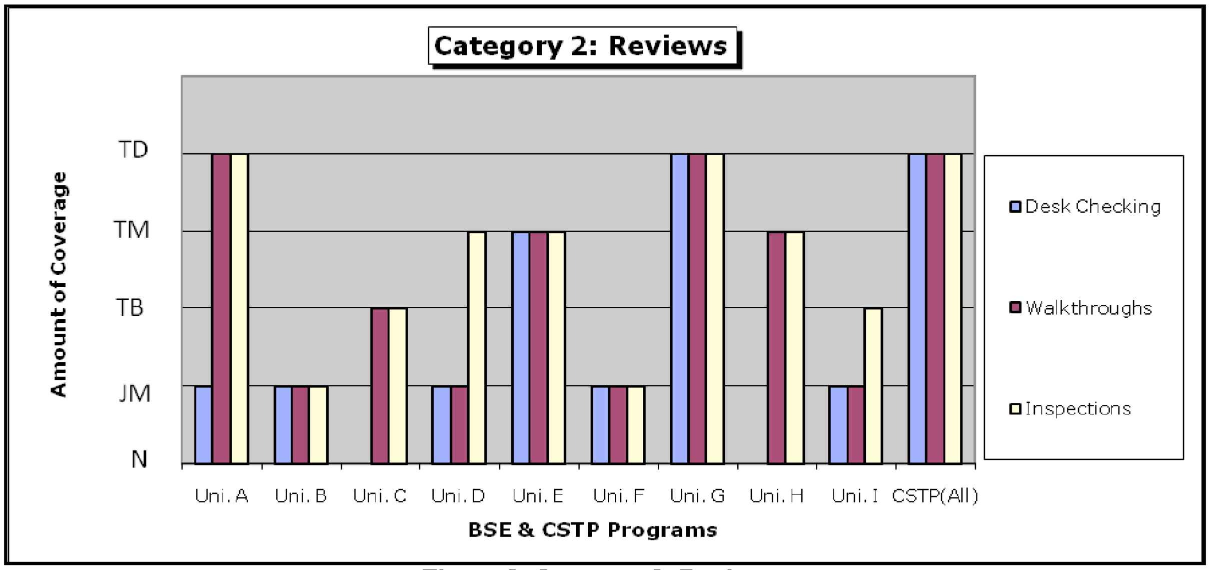

Figure 2. Category 2: Reviews.

Category 2 consists of three main topics: desk checking, walkthroughs, and inspections. The suggested number of hours for this category in the ACM Guidelines is six. The CSTP program and University G covered all three topics in depth. Two universities (B and F) just mentioned these topics. Inspection was covered by most of the other universities. The second topic with a coverage close to that of inspection was walkthroughs, and the least emphasized topic was desk checking.

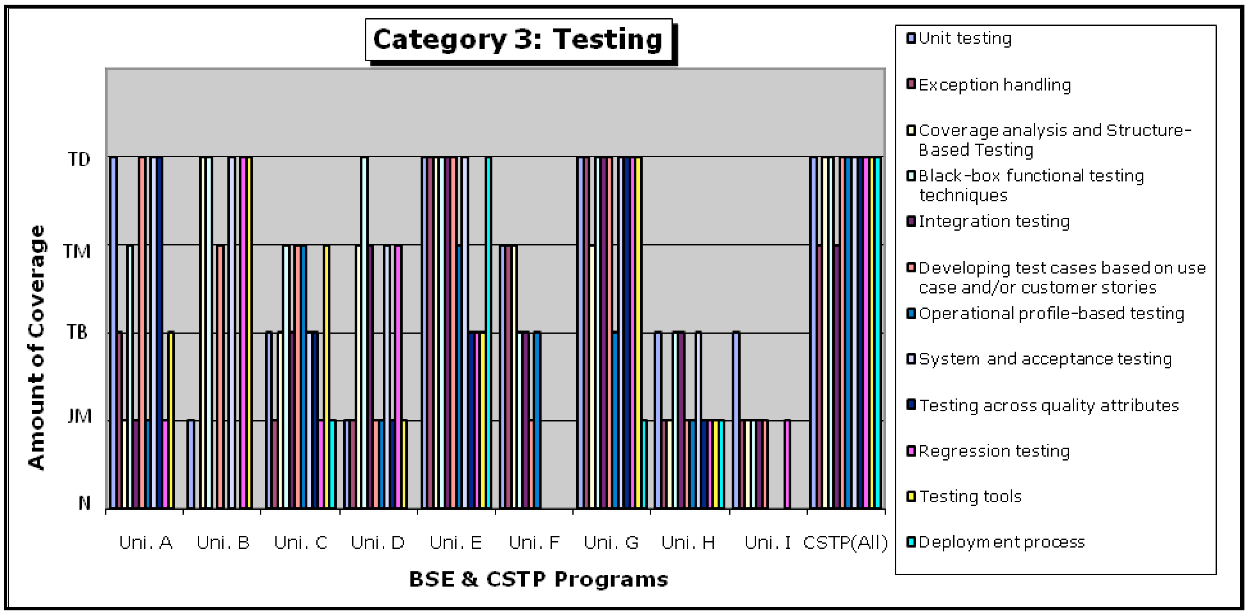

Figure 3. Category 3: Testing.

Category 3 involving 12 topics was the most important category in the ACM Curriculum Guidelines [2], with 21 hours suggested for its coverage. The CSTP program covered all topics in depth, except for the two topics of exception handling and integration testing. Topics that were significantly different in terms of their coverage by the universities compared to the CSTP program were unit testing, operational profile-based testing, testing across quality attributes (for example, usability, security, compatibility, and accessibility) and testing tools. The topic with the largest difference in coverage was the deployment process.

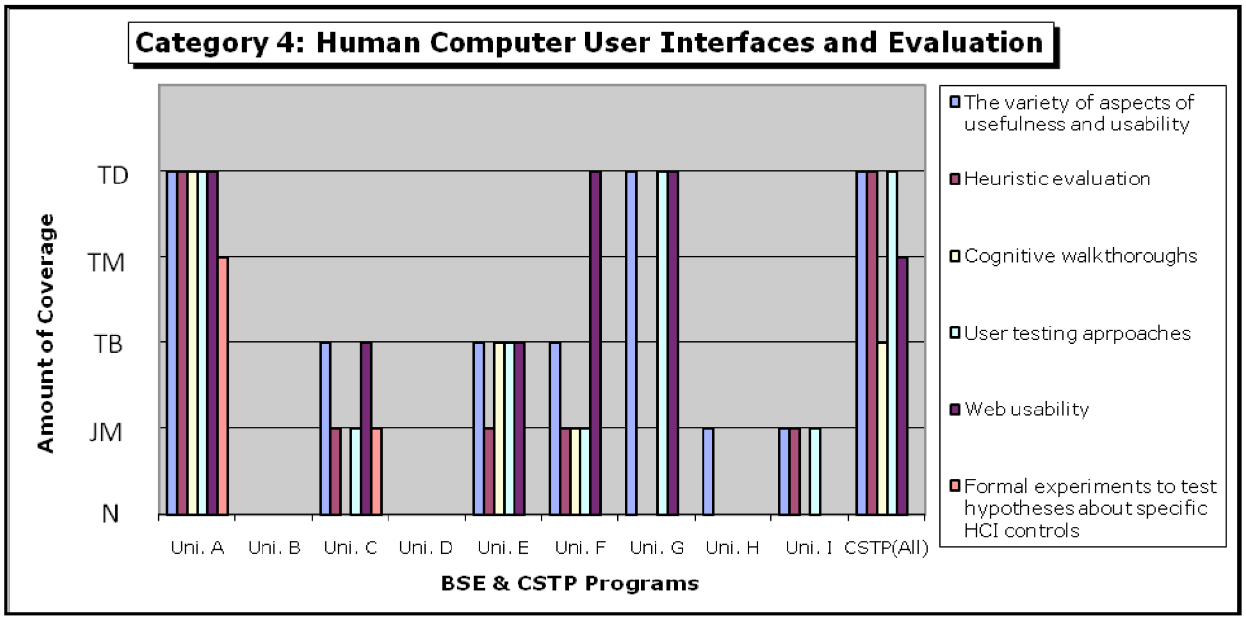

Figure 4. Category 4: Human computer user interfaces and evaluation.

Category 4 is considered to be the least emphasized category in most universities because its topics are not frequently included in the courses offered. This category consists of six topics with the six hours of coverage suggested in the ACM Guidelines. Significant gaps in coverage existed in the case of Universities B and D with none of the topics included. Only three universities (A, C, and E) covered all the topics, albeit to different extents. The topic that had least emphasis was that dealing with formal experiments to test the hypotheses about specific HCI controls. This topic was not taught at all in the CSTP program. The CSTP program covered three topics in depth: the variety of aspects of usefulness and usability, heuristic evaluation, and user testing approaches, while cognitive walkthroughs was least emphasized and was not taught in six universities (C, E, F, G, H, and I).

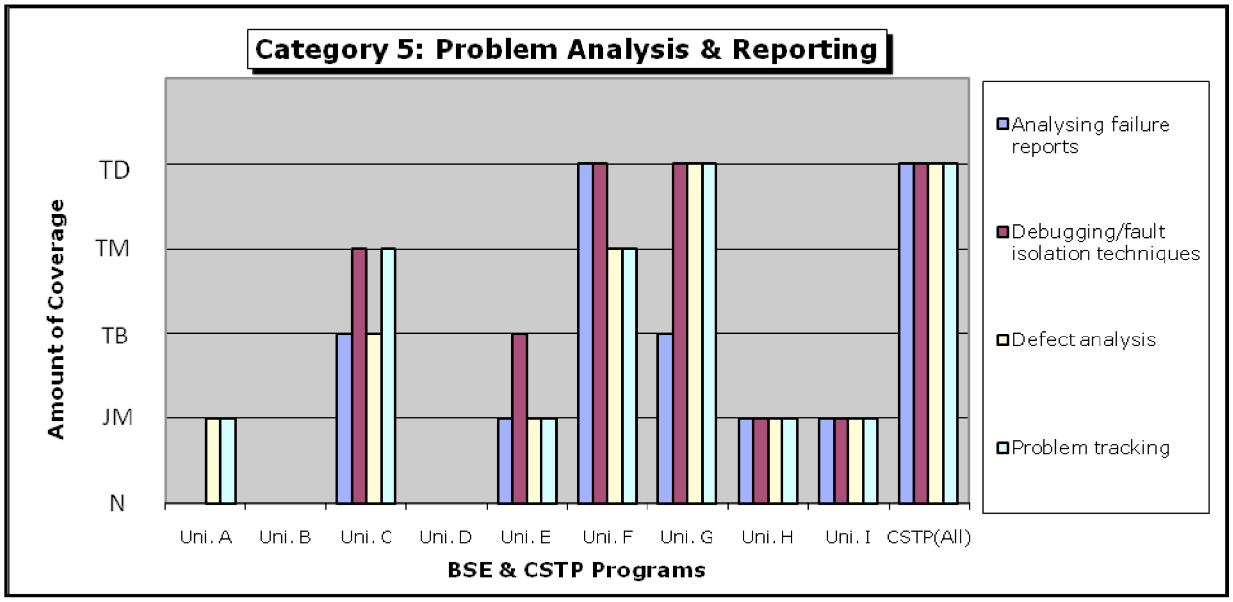

Figure 5. Category 5: Problem analysis and reporting.

Category 5 consists of four topics requiring four hours of coverage according to the ACM Guidelines: analyzing failure reports, debugging/fault isolation techniques, defect analysis, and problem tracking. Significant gaps in the coverage here existed in five universities. Two universities (B and D) did not teach all topics, while three universities (A, H, and I) only offered a mention of the topics. The CSTP program taught these topics in depth. The topic that had least emphasis, analyzing failure reports, had just a mention in four universities (E, G, H, I, and J), while debugging/fault isolation techniques and problem tracking were covered beyond the basic level in three universities (C, F, and G).

Based on the above analysis for each category, there are 11 topics that are most significant in that they receive the least emphasis in BSE programs (see Table 3 below). Category 3 (Testing), the most essential category in Software V&V Knowledge, shows the highest number of least emphasized topics compared to other categories. The analysis thus shows that there is an important need to enhance the extent of teaching devoted to unit testing in BSE programs. Additionally, this result is significant since the survey of Australian industry requirements shows that unit testing is an extensively used and essential skill for software testers [6], [24].

Table 3. The significant least emphasis topics in BSE program

Category	Topics
1	• V&V involvement at different points in the lifecycle
2	• Desk checking
3	• Unit testing • Operational profile-based testing • Testing across quality attributes • Testing tools • Deployment process
4	• Cognitive walkthroughs
5	• Analyzing failure reports • Debugging/fault isolation techniques • Problem tracking

4.1.2. Part Two: Methods of teaching

The six methods of teaching that have been used comprise lectures, tutorials, practical work, assignments, projects, and examinations. Four methods, namely tutorials, practical work, assignments, and projects varied in their usage among the universities. For example, while tutorials and practical work sometimes involved the same tasks, in other universities, these methods could comprise different tasks. Therefore, such methods of teaching are considered here as hands-on activities which can be combined into one method referred to as 'Practicals' (P). On the other hand, lectures as a theoretical activity (L) comprise a stand-alone method. The aim of so grouping these methods was to enable the examination of future methods of teaching and learning to involve two parts only, one lecture-based and one practical-based.

Table 4 shows that for Category 4 (Human computer user interfaces and evaluation) and Category 5 (Problem analysis and reporting), two universities only used lectures to teach the topics under each category. The other categories did not show significant differences.

Table 4. Methods of teaching by university and CSTP Program

Universities & CSTP	Uni. A		Uni. B		Uni. C		Uni. D		Uni. E		Uni. F		Uni. G		Uni. H		Uni. I		CSTP Program	
Methods	L	P	L	P	L	P	L	P	L	P	L	P	L	P	L	P	L	P	L	P
Category 1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Category 2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√
Category 3	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Category 4	√	√	√	√	√	-	-	-	√	√	√	√	√	√	√	√	√	-	√	√
Category 5	√	-	√	√	√	√	-	-	√	√	√	√	√	√	√	-	-	-	√	√

4.2. Challenges to the validity of the study

The survey conducted here faces possible challenges that may arise when it is compared to other experiments and case studies [25], even if it was planned with an adequate consciousness of such challenges. There exist three types of potential challenges [26], those relating to internal validity, to construct validity, and to external validity. A fourth validity concern, conclusion validity, is not included because it can be clearly considered only when appropriate statistical conclusions are drawn. In the present survey, this was not the case, especially in Part Two (Methods of Teaching), because the outcomes were of a more general nature. The ensuing discussion considers the first three challenges cited.

4.2.1. Internal validity

There are two issues that arise in relation to internal validity. The first issue concerns the survey results that involved only one answer from a tutor who assisted a lecturer for one course, while other course data came from lecturers who were experienced teachers in these courses. However, as the tutor taught the course for more than one semester, it is considered that the answer is acceptable.

The second issue related to one course which was taught by two lecturers in the same semester. Lecturer A taught from Week 1 until Week 7 and Lecturer B from Week 8 until Week 13. Table 5 shows that for Category 1, after a follow-up discussion, the responses from Lecturer A appeared incomplete. The reason was that Lecture A did not tick the right answer for this category because he overlooked it. There was also some confusion about practical work in this course in that it was considered either as assignments or as a project. This probably explains the differences revealed for Category 3. Only Lecturer B considered that the topics in Category 4 were to be taught. In fact, the internal validity threat can affect the independent variable with respect to causality without the researcher's knowledge [26]. For instance, where Lecturer A causes decisions or actions by Lecture B, or otherwise. Therefore, in this case, the same course taught by two lecturers engendered slight misunderstandings concerning methods of teaching.

Table 5. Methods of teaching for two lecturers

Methods	Category 1	Category 2	Category 3	Category 4	Category 5
Lecture	Both	-	Both	Lecturer B	-
Tutorial	Both	-	Both	-	-
Practical	Both	-	-	-	-
Assignment	Lecturer B	-	Lecturer A	-	-
Exam	-	-	Both	-	-
Project	Lecturer B	-	Lecturer B	Lecturer B	-
Other	-	-	-	-	-

4.2.2. External validity

In this study, the validity of the online survey suffered because of the low response rate [22]. Of the 16 universities targeted, only nine responded (56.25%). Three lecturers in the other universities were reluctant to participate, while 14 lecturers in five of the universities approached did not respond. Reminder messages were sent where required to try to ensure that the survey obtained sufficient data. Despite these challenges, it is considered that the survey data provided the information required to reveal the current state-of-practice in teaching Software V&V in Australian universities.

4.2.3. Construct validity

Here, the potential for false responses from the participants was considered. For instance, University G covers in depth most of Software V&V topics in the four categories 1, 2, 3, and 5. In this case, those surveyed could have provided information that boosted the academic appearance of their program. Unfortunately, such an issue cannot be analyzed through the survey data.

5. Future Work

To become effective software testers, students should be able to think technically, creatively, critically, and practically [27]. Consequently, methods of teaching and learning that should be examined further for their adequacy in these respects are those that are more practical-based. As is well known, problem-based learning (PBL) is a method in which learners on first encountering a problem undertake a systematic, student-centered enquiry process [28]. Ideally, PBL encourages problem-solving skills, cooperative learning, independent work, active engagement among students, and other such means that can encourage BSE students to work in a team and think like real software testers.

More challenging Software V&V activities can be designed for inclusion in PBL project-based courses with experienced software testers as external evaluators working alongside the course instructors. Additionally, such industry-based courses can provide BSE graduate students with professional and real-world training in software testing. In fact, the course could be designed as a project-based course that incorporates the various Software V&V topics, including, for instance, V&V involvement at different points in the software lifecycle, unit testing, testing tools, analyzing failure reports, and debugging techniques. One further issue concerns avoiding offering new Software V&V courses within already over-packed curricula [9]. Accordingly, such a course should be targeted only at those BSE graduate students who are interested in becoming qualified software testers.

Moreover, to ensure the use of learning styles that are more efficient, the Pex tool [29] based on C# language is identified as an appropriate tool for supporting the teaching of Software V&V topics such as performing unit testing, analyzing failure reports, and debugging. This tool is capable of producing traditional unit test cases automatically with high code coverage, suggesting a bug fix when a generated test fails, and generating the HTML reports that contain, inter alia, generated tests, path conditions, and suggested fixes. Indeed, a series of empirical studies should be designed to analyze the effectiveness of methods of teaching and learning, testing tools, and testing activities for teaching Software V&V topics. This will contribute to ensuring that BSE students are equipped with the necessary testing skills, methodologies, and knowledge that are in line with industry needs. These should include, for example, defining test objectives, designing test cases, documenting test results, reusing the same test cases after changes were made to the software, and redesigning test cases based on the analysis of previous test results [6].

6. Conclusions

Overall, the above analysis comparing university BSE programs and the CSTP program for each category of Software V&V Knowledge shows that differences exist in the extent of the coverage of the topics involved. The most significant least emphasized topics are V&V involvement at different points in the software lifecycle, desk checking, unit testing, operational profile-based testing, testing quality attributes, testing tools, deployment process, cognitive walkthroughs, analyzing failure reports, debugging or fault isolation techniques, and problem tracking. In addition, with respect to methods of teaching for Category 4 (Human Computer User Interfaces and Evaluation) and Category 5 (Problem Analysis and Reporting), both applied more lecture-based sessions than practical-based sessions. However, most universities mixed both these methods of teaching for Category 1 (V&V Terminology and Foundations), Category 2 (Reviews), and Category 3 (Testing).

The survey results elicited here indicate that it is important to enhance the learning and teaching coverage of Software V&V topics. This is especially apparent, for instance, in the case of unit testing that is commonly applied in industry [8] and also in the case of topics that reflect other industry needs [6]. The suggested PBL industry project-based course which blends with challenging software V&V testing activities and the use of the Pex tool (to encourage TDD practice) is to be targeted so as to attract those BSE graduate students who are interested in furthering their careers as software testers. This course aims at combining the five least emphasized topics cited in the Future Work section above. The effectiveness of such a course can be examined with a series of empirical studies that have a qualified and experienced software tester as an external evaluator.

7. Acknowledgements

I would like to thank all who participated in the survey, BSE coordinators and lecturers from nine Australian universities. I would also like to express my gratitude to Dr Ross from K.J Ross & Associates, for giving me the opportunity to sit in on CSTP Foundation Level Course and to Mr. Emmett for providing me with valuable information with regard to the other CSTP courses.

8. References:

[1] G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, The art of software testing, 2nd ed. Hoboken, N.J.: John Wiley & Sons, 2004.

[2] Software Engineering 2004, "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering," in Available online at :http://sites.computer.org/ccse/SE2004Volume.pdf, 2004.

[3] E. L. Jones and C. L. Chatmon, "A perspective on teaching software testing," presented at Proceedings of the seventh annual consortium for computing in small colleges central plains conference on The journal of computing in small colleges, Branson, Missouri, United States, 2001.

[4] T. Shepard, M. Lamb, and D. Kelly, "More testing should be taught," Communications of the ACM, vol. 44, pp. 103-108, 2001.

[5] P. Bourque, F. Robert, J. M. Lavoie, A. Lee, S. Trudel, and T. C. Lethbridge, "Guide to the Software Engineering Body of Knowledge (SWEBOK) and the Software Engineering Education Knowledge (SEEK) - a preliminary mapping," 2002.

[6] S. P. Ng, T. Murnane, K. Reed, D. Grant, and T. Y. Chen, "A Preliminary Survey on Software Testing Practices in Australia," presented at Proceedings of the 2004 Australian Software Engineering Conference (ASWEC), 2004.

[7] "Certified Software Test Professional (CSTP) Programme, Handbook 2006," Release V9.0 ed: ObjectTraining a division of Object Consulting, Available at: http://www.kjross.com.au/page/Training/Certified_Software_Test_Professional, March 2006.

[8] M. A. Wojcicki and P. Strooper, "A state-of-practice questionnaire on verification and validation for concurrent programs," presented at Proceeding of the 2006 Workshop on Parallel and Distributed Systems: Testing and Debugging, Portland, Maine, USA, 2006.

[9] S. Elbaum, S. Person, J. Dokulil, and M. Jorde, "Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable," 2007.

[10] T. Y. Chen and P. Pak-Lok, "Experience with teaching black-box testing in a computer science/software engineering curriculum," IEEE Transactions on Education, vol. 47, pp. 42-50, 2004.

[11] K. Fereydoun and H. Trudy, "A software testing course for computer science majors," SIGCSE Bull., vol. 37, pp. 50-53, 2005.

[12] D. Hoffman, P. Strooper, and P. Walsh, "Teaching and testing," presented at Proceedings of the Ninth Conference on Software Engineering Education, 1996.

[13] C. Kaner, "Teaching domain testing: a status report," presented at Proceedings of the 17th Conference on Software Engineering Education and Training, 2004.

[14] S. Padmanabhan, "Domain Testing: Divide and Conquer," in Computer Engineering, Master Thesis: Florida Institute of Technology, 2004, pp. 131.

[15] A. Tinkham and C. Kaner, "Experiences teaching a course in programmer testing," presented at Proceedings of the Agile Development Conference, 2005.

[16] E. L. Jones, "Integrating testing into the curriculum \— arsenic in small doses," presented at Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, Charlotte, North Carolina, United States, 2001.

[17] C. Depradine and J. Arthur, "A tool for incorporating unit testing into a Java programming curriculum," presented at Proceedings of the 3rd international symposium on Principles and practice of programming in Java, Las Vegas, Nevada, 2004.

[18] S. H. Edwards, "Using software testing to move students from trial-and-error to reflection-in-action," presented at Proceedings of the 35th SIGCSE technical symposium on Computer science education, Norfolk, Virginia, USA, 2004.

[19] A. Patterson, "Tool Support for Introductory Software Engineering Education," in School of Computer Science and Software Engineering, PhD thesis. Melbourne, Australia: Monash University, 2002, pp. 158.

[20] E. Allen, R. Cartwright, and B. Stoler, "DrJava: a lightweight pedagogic environment for Java," presented at Proceedings of the 33rd SIGCSE technical symposium on Computer science education, Cincinnati, Kentucky, 2002.

[21] N. Tillmann and J. de Halleux, "White-box testing of behavioral web service contracts with Pex," presented at Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications, Seattle, Washington, 2008.

[22] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research part 2: designing a survey," SIGSOFT Softw. Eng. Notes, vol. 27, pp. 18-20, 2002.

[23] B. Kitchenham and S. L. Pfleeger, "Principles of survey research: part 5: populations and samples," SIGSOFT Softw. Eng. Notes, vol. 27, pp. 17-20, 2002.

[24] M. A. Wojcicki and P. Strooper, "A state-of-practice questionnaire on verification and validation for concurrent programs," in Proceedings of the 2006 workshop on Parallel and distributed systems: testing and debugging. Portland, Maine, USA: ACM, 2006.

[25] T. Punter, M. Ciolkowski, B. Freimut, and I. John, "Conducting on-line surveys in software engineering," 2003.

[26] C. Wohlin, Experimentation in Software Engineering:: an Introduction: Springer, 1999.

[27] C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing: John Wiley & Sons, Inc. New York, NY, USA, 2001.

[28] P. L. Schwartz, G. Webb, and S. Mennin, Problem-Based Learning: Case Studies, Experience and Practice: Routledge (UK), 2001.

[29] Microsoft Corporation, "pex v1.1.20813.0 Test Generation for .NET," 2007.

APPENDIX

Confirmation for Ethical Review:

The University of Queensland
Brisbane Qld 4072 Australia
Telephone +61 7 3365 2097
Facsimile +61 7 3365 4999
Email enquiries@itee.uq.edu.au
Internet www.itee.uq.edu.au

School of Information Technology and Electrical Engineering

HEAD OF SCHOOL
Professor Paul Bailes

A State-of-Practice on Teaching Software Testing in Software Engineering Program for Australian Undergraduate Studies.

Information Sheet

In this survey you will have to answer thirty (30) questions based on the ACM Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering: http://sites.computer.org/ccse/SE2004Volume.pdf . This survey is divided into the following five categories:

a) V&V Terminology and Foundations
b) Reviews
c) Testing
d) Human Computer User Interfaces and Evaluation
e) Problem Analysis and Reporting

We are interested in how well the teaching testing performs mainly the content of software verification and validation is being taught in Bachelor of Software Engineering program for Australian undergraduate studies.

All data recorded in this survey will be recorded and stored with an anonymous subject identifier which will allow us to record specific information: a) the coverage of teaching testing, b) to identify the method of teaching and c) assessment which reflect to five categories mentioned above. You can withdraw from the study at any point without prejudice or penalty of any kind.

The survey will run for approximately thirty to one hour. There are no foreseeable added risks to you above the risks of everyday living. Before proceeding further, we need you to read the instructions provided in the questionnaire.

This study has been cleared in accordance with the ethical review processes of the University of Queensland. You are, of course, free to discuss your participation with project staff (contactable on: zulfa@itee.uq.edu.au for Zulfa Zakaria). If you would like to speak to an officer of the University not involved in the study, you may contact the School of Information Technology and Electrical Engineering Ethics Officer directly on 3365 3476, or contact the University of Queensland Ethics Officer on 3365 3924.

Project Staff:
Zulfa Zakaria
School of Information Technology and Electrical Engineering
General Purpose South
University of Queensland, QLD 4072
Tel: x51136
Email: zulfa@itee.uq.edu.au

Sample of Online Questionnaire:

Title of Research: A Survey on Teaching Software Testing in Australian Undergraduate Software Engineering Programs.

The purpose of this survey is to get information from lecturers who are teaching Software Validation and Verification (V&V) in Bachelor of Software Engineering programs. This survey will compare the content of courses covering Software V&V with the ACM Curriculum Guidelines for Australian Undergraduate Degree Programs in Software Engineering: http://sites.computer.org/ccse/SE2004Volume.pdf. This survey is divided into FIVE (5) categories:

a) V&V Terminology and Foundations
b) Reviews
c) Testing
d) Human Computer User Interfaces and Evaluation
e) Problem Analysis and Reporting

Note that, an additional category is added for seeking information about Extreme Programming (XP) and Test-driven Development (TDD). This category is called Category 6: Development Paradigms.

We assure that all survey data, comments and responses are anonymous and will be treated as strictly confidential. The collected material will be accessed by the research team only. The aggregated results will be anonymous, therefore, there will be no identification of individual programs. The information gathered will be used for statistical purposes and help us to set the direction for future research. In addition, each respondent will be provided with a copy of the report from this survey and any publications that may result from it.

This study has been cleared in accordance with the ethical review processes of the University of Queensland. You are, of course, free to discuss your participation with project staff (contactable on: zulfa@itee.uq.edu.au for Zulfa Zakaria). If you would like to speak to an officer of the University not involved in the study, you may contact the School of Information Technology and Electrical Engineering Ethics Officer directly on 07- 3365 3476, or contact the University of Queensland Ethics Officer on 07- 3365 3924

INSTRUCTION FOR THE SURVEY QUESTIONNAIRE

Each category consists of the following parts:

Part i) Amount of coverage:
This part is to get information about how much teaching is done for all the topics in each category. There are FIVE (5) possible answers:

Codes	Explanation
N	None
JM	Just a mention
TB	Teach the basics (1-2 hours)
TM	Teach moderately (2-3 hours)
TD	Teach in depth (more than 3 hours)

Part ii) Method of teaching:
This part is to gain information about how each element of testing is taught: as a lecture, practical work, tutorial, assignment, exam or project. For this part, you can select more than one element

Click here for **Glossary**

LECTURER AND COURSE DETAILS:

Lecturer's Name:

Contact Number: **Email:**

University's Name:

Course's Name: **Course Code:**

CATEGORY 1: VERIFICATION & VALIDATION (V&V) TERMINOLOGY AND FOUNDATIONS.

Part i) **Amount of coverage**: Tick **ONE (1)** code only for each topic below:

Codes refer to the number of hours	**N**	**JM**	**TB**	**TM**	**TD**
A1) Objectives and constraints of V&V	○	○	○	○	○
A2) Planning the V&V effort	○	○	○	○	○
A3)Documenting V&V strategy (including tests and other artifacts)	○	○	○	○	○
A4)Metrics & measurement (e.g. reliability, usability, performance etc.)	○	○	○	○	○
A5) V&V involvement at different points in the lifecycle	○	○	○	○	○

Part ii) For the topics in category 1, what **method of teaching** do you use? (Tick all that apply)

☐ Lecture

☐ Tutorial

☐ Practical

☐ Assignment

☐ Exam

☐ Project

☐ Other, please specify:

CATEGORY 2: REVIEWS.

Part i) **Amount of coverage** : Tick **ONE (1)** code only for each topic below:

Codes refer to the number of hours	**N**	**JM**	**TB**	**TM**	**TD**
B1) Desk checking	○	○	○	○	○
B2) Walkthroughs	○	○	○	○	○
B3) Inspections	○	○	○	○	○

Part ii) For the topics in category 2, what **method of teaching** do you use? (Tick all that apply)

☐ Lecture

☐ Tutorial

☐ Practical

☐ Assignment

☐ Exam

☐ Project

☐ Other, please specify:

CATEGORY 3: TESTING.

Part i) **Amount of coverage:** Tick **ONE (1)** code only for each topic below:

Codes refer to the number of hours	**N**	**JM**	**TB**	**TM**	**TD**
C1) Unit testing	○	○	○	○	○
C2) Exception handling (writing test cases to trigger exception handling; designing good handling)	○	○	○	○	○
C3) Coverage analysis and structure based testing (e.g. statement, branch, basis path, multi-condition, dataflow, etc.)	○	○	○	○	○
C4) Black-box functional testing techniques	○	○	○	○	○
C5) Integration testing	○	○	○	○	○
C6) Developing test cases based on use cases and/or customer stories	○	○	○	○	○
C7) Operational profile-based testing	○	○	○	○	○
C8) System and acceptance testing	○	○	○	○	○
C9) Testing across quality attributes (e.g. usability, security, compatibility, accessibility, etc.)	○	○	○	○	○
C10) Regression testing	○	○	○	○	○

C11) Testing tools	○	○	○	○	○
C12) Deployment process	○	○	○	○	○

Part ii) For the topics in category 3, what **method of teaching** do you use? (Tick all that apply)

☐ Lecture

☐ Tutorial

☐ Practical

☐ Assignment

☐ Exam

☐ Project

☐ Other, please specify:

CATEGORY 4: HUMAN COMPUTER USER INTERFACES AND EVALUATION.

Part i) **Amount of coverage:** Tick **ONE (1)** code only for each topic below:

Codes refer to the number of hours	**N**	**JM**	**TB**	**TM**	**TD**
D1) The variety of aspects of usefulness and usability	○	○	○	○	○
D2) Heuristic evaluation	○	○	○	○	○
D3) Cognitive walkthroughs	○	○	○	○	○
D4) User testing approaches (observation sessions etc.)	○	○	○	○	○
D5) Web usability (testing techniques for web sites)	○	○	○	○	○
D6) Formal experiments to test hypotheses about specific HCI controls	○	○	○	○	○

Part ii) For the topics in category 4, what **method of teaching** do you use? (Tick all that apply)

☐ Lecture

☐ Tutorial

☐ Practical

☐ Assignment

☐ Exam

☐ Project

☐ Other, please specify:

CATEGORY 5: PROBLEM ANALYSIS & REPORTING.

Part i) **Amount of coverage:** Tick **ONE (1)** code only for each topic below:

Codes refer to the number of hours	**N**	**JM**	**TB**	**TM**	**TD**
E1) Analysing failure reports	○	○	○	○	○
E2) Debugging / fault isolation techniques	○	○	○	○	○
E3) Defect analysis	○	○	○	○	○
E4) Problem tracking	○	○	○	○	○

Part ii) For the topics in category 5, what **method of teaching** do you use? (Tick all that apply)

☐ Lecture

☐ Tutorial

☐ Practical

☐ Assignment

☐ Exam

☐ Project

☐ Other, please specify:

CATEGORY 6: Development Paradigms.

Part i) **Amount of coverage:** Tick **ONE (1)** code only for each question below:

Codes refer to the number of hours	**N**	**JM**	**TB**	**TM**	**TD**
F1) Extreme programming (XP)	○	○	○	○	○
F2) Test-Driven development (TDD)	○	○	○	○	○

Part ii) For the topics in category 6, what **method of teaching** do you use? (Tick all that apply)

☐ Lecture

☐ Tutorial

☐ Practical

☐ Assignment

☐ Exam

☐ Project

☐ Other, please specify:

General Comments
Any comments or additional information you want to add about this questionnaire?

Submit

Results for University A:

TOPICS	Course 1(R)	Course 2(R)	Course 3(R)	Course 4(E)	Course 5(E)	University A
CATEGORY 1: V&V TERMINOLOGY AND FOUNDATIONS						
1. Objectives and constraints of V&V	TB	JM	JM	N	TM	TD
2. Planning the V&V effort	JM	JM	JM	N	TD	TD
3. Documenting V&V strategy (including tests and other artifacts)	TB	JM	JM	N	TD	TD
4. Metrics & Measurement (e.g. reliability, usability, performance etc.)	N	N	TB	N	TD	TD
5. V&V involvement at different points in the lifecycle	JM	JM	JM	N	JM	JM
CATEGORY 2: REVIEWS						
6. Desk Checking	N	N	JM	N	TM	TM
7. Walkthroughs	N	N	JM	N	TD	TD
8. Inspections	N	N	TM	N	TD	TD
CATEGORY 3: TESTING						
9. Unit testing	TB	TM	JM	TM	N	TD
10.Exception handling (writing test cases to trigger exception handling; designing good handling)	JM	TB	N	N	N	TB
11. Coverage analysis and Structure-Based Testing (e.g. statement, branch, basis path, multi-condition, dataflow, etc.)	JM	JM	JM	N	N	JM
12.Black-box functional testing techniques	JM	JM	N	TM	N	TM
13.Integration testing	JM	JM	JM	N	N	JM
14.Developing test cases based on use case and/or customer stories	JM	JM	JM	N	TD	TD
15.Operational profile-based testing	N	JM	N	N	N	JM
16.System and acceptance testing	JM	JM	JM	N	TD	TD
17. Testing across quality attributes (e.g. usability, security, compatibility, accessibility, etc.)	N	N	TB	N	TD	TD
18.Regression testing	JM	JM	JM	N	N	JM
19.Testing tools	JM	TB	JM	JM	N	TB
20.Deployment process	N	N	N	N	N	N
CATEGORY 4: HUMAN COMPUTER USER INTERFACES AND EVALUATION						
21.The variety of aspects of usefulness and usability	N	JM	JM	N	TD	TD
22.Heuristic evaluation	N	N	N	N	TD	TD
23.Cognitive walkthroughs	N	N	N	N	TD	TD
24. User testing approaches (observation sessions etc.)	N	N	N	N	TD	TD
25.Web usability (testing techniques for web sites)	N	N	N	N	TD	TD
26.Formal experiments to test hypotheses about specific HCI controls	N	N	N	N	TM	TM
CATEGORY 5: PROBLEM ANALYSIS & REPORTING						
27.Analyzing failure reports	N	N	N	N	N	N
28.Debugging/fault isolation techniques	N	N	N	N	N	N
29.Defect analysis	N	JM	N	N	N	JM
30.Problem tracking	N	N	JM	N	N	JM
CATEGORY 6: DEVELOPMENT PARADIGMS						
31.Extreme Programming (XP)	N	JM	TB	N	N	TB
32.Test-Driven Development (TDD)	N	N	JM	N	N	JM

Note that R and E in the bracket for each course is stand for required courses and E for elective courses.