Investigation of software tester responsibilities, personality and performance

by

Tanjila Kanij

A Thesis Submitted for the Degree of

Doctor of Philosophy

at Faculty of Information and Communication Technologies

Swinburne University of Technology

John Street, Hawthorn- 3122

Australia

I would like to dedicate this work to my caring parents and my loving husband. I hope that this work makes them proud.

Abstract

Software testing refers to the process of reviewing or executing software with a view to detecting faults before delivery. The successful accomplishment of the process includes carrying out a number of unit tasks. A refined list of such unit tasks is not clear. However, the goal of the role indicates testing comprises of a fundamentally different task set to those of other software development practitioners. This raises the question of whether the personality of software testers is different to other people involved in software development. These personality traits may also influence the performance in this role. However a generalized method for assessing the performance of software testers is also unavailable.

This research aims to address some of these issues. The research started with finding the factors that affect software testers' performance. We collected information from different sources to prepare a list of unit tasks of software testing . We also collected big five factor based personality profiles of software practitioners including testers to investigate if software testers differed from others. To find the connection of the personality traits on the effectiveness in software testing we conducted a research study with student participants. Finally we proposed a novel performance appraisal form for software testers and collected feedback on the proposed form from software development project managers.

Our findings indicate that along with test specific activities software testers perform a number of unit tasks such as “Analysing requirements and functional design”, “Debugging” and “Maintaining test infrustructure”. Software testers are generally higher on conscientiousness compared to other related practitioners. Conscientiousness is also weakly related to bug location rate and weighted fault density which are used as measures of testing performance.

Acknowledgements

I take the privilege to express my sincere gratitude to my supervisors Prof. John Grundy and Dr. Robert Merkel for their guidance, support and patience. The completion of this thesis would not be possible without my kind mentors.

I am grateful to my loving husband for his active support and encouragement throughout the journey of this thesis.

My special thanks go to my mum Hosne Ara Hossain, my mother-in-law Monoara Ahmed and elder sister Nusrat Zaman for their unconditional family assistance during my thesis. I also thank my one year old baby girl for being supportive!

I also wish to thank my dear friends Shimul, Arup, Shibli, Tanay, Samiran, Galib, Imrul, Saiful, Mohammed, Amani, Feifei and Khalid for their help. My research works would be somewhat incomplete without their kind response to my request of participation in all my pilot projects.

Declaration

I herewith declare that I have produced this thesis with my own work without the prohibited assistance of third parties. This study has not previously been presented as a thesis.

Tanjila Kanij,
Dated

The Author's Publications

1. Tanjila Kanij, Robert Merkel, John Grundy: A Preliminary Study on Factors Affecting Software Testing Team Performance. ESEM 2011: 359-362
2. Tanjila Kanij, Robert Merkel, John Grundy: Performance assessment metrics for software testers. CHASE 2012: 63-65
3. Tanjila Kanij, Robert Merkel, and John Grundy, An empirical study of the effects of personality on software testing, Accepted to be published at 26th Conference on Software Engineering Education and Training, 2013.
4. Tanjila Kanij, Robert Merkel, and John Grundy, Lessons learned from conducting industry surveys in software testing, Accepted to be published at First International Workshop on Conducting Empirical Studies in Industry, CESI 2013.
5. Tanjila Kanij, Robert Merkel, and John Grundy, Performance appraisal of softwar testers, Submitted for review at the Journal of Information Science and Technology, special edition, 2013.
6. Robert Merkel and Tanjila Kanij, Does the individual matter in software testing? http://www.swinburne.edu.au/ict/research/sat/-technicalReports/TC2010-001.pdf.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

This thesis describes our research to address different issues in software testing. These include the understanding of the unit tasks performed in this role, human factors influencing performance in this role, effect of personality traits on software testing performance and better understanding of performance appraisal of software testers. This chapter presents the background and motivation of this research followed by problem statement and a brief description to the research. The significance of this research is also illustrated in this chapter. The chapter is concluded with the organization of the rest of the thesis.

1.2 Background

Software testing is an essential part of software development. It refers to the activity to verify the developed software meets expected requirements. The software testing activities are not necessarily restricted to executing the developed software to identify defects in it, the process also includes planning, designing, reporting, reviewing, ensuring defects are fixed and so on. The activities are primarily important to guarantee that the software satisfies the agreed upon requirements. This is also important to develop and deliver reliable and better quality software. The software quality is attributed to the factors such as correctness, reliability, usability, maintainability, reusability, testability, and so on. If testing is performed well in development of the software, the developing organization can gain certain level of confidence on the delivered software product which also reduces their maintenance budget.

Due to the many benefits of software testing, it has become a business industry itself. According to an article published in “RedOrbit.com” on 11th March, 2009, testing service will continue to rise at a rate of 9.5% from 2008 to 2013 and will be an $56 billion industry by 2013 [1]. Software testers are key players in the testing groups, companies and the testing industry as a whole. From analysing requirements and developing and/or selecting best approach for testing to interpreting and communicating the test results, software testers play the key role. According to Mr Kris Gopalakrishnan, the Infosys CEO, due to the increased responsibilities of software testers and the importance of their role on developing reliable software, software testers have become essential part of software development management. In his own words- “*Today the tester has won a seat on the product management table*” [2].

1.3 Research Motivation

A software tester is a person who tests software before release and helps to increase the reliability of the software product by reporting bugs and getting them fixed. According to the text books on software testing and the job advertisements we see that software testing responsibilities include tasks such as “test planning”, “test execution”, “reporting the results”, “bug advocacy” and so on. There is however no empirically validated list of unit tasks performed as part of software testing. Although this list may not always be static, such a list is important to get an idea of this role. This lack of clear information about software testers’ job responsibilities impedes research on finding who is best suited for this role. For example it is impossible to assess who is best suited for the role of software tester without knowing what testers actually do.

According to the Guide to the Software Engineering Body of Knowledge [3], testing related activities are classified in seven broad categories: planning, Test-case generation, test environment development, execution, test results evaluation, problem reporting/Test log and defect tracking. However, what the unit tasks are that make up each of these broad categories of actives is not clear. Although a compiled list of the unit tasks of this role is unavailable, from the available information it is clear that this role is different to other practitioners involved in the software development process. While software programmers are largely constructive, in that they design and "build" something that meets customer requirements, testers' job is in a sense destructive in that they attempt to "break" the software constructed by programmers. This needs a destructive approach towards the module to be tested. The destructive approach is to construct best quality product possible. Thus it is constructive in a broader sense. The alternative approach of the profession raises a question whether certain human characteristics have any influence on the effectiveness in this role. Those human characteristics might also be different to other software development practitioners.

According to Weinberg [4] and Pressman [5] the performance of software engineers is somehow dependent on their individual characteristics. Pressman said it is "innate human traits", and Weinberg suggested that it is "Combinations of skills and personality traits" that influence the effectiveness of a software engineer. However, little research has been conducted on the influence of human characteristics on the effectiveness of software testers. The majority of software testing research has been devoted to the enhancement of testing processes, test criteria, and to the development of new techniques and tools for different types of testing [6]. The trend of the research in this field implies a key assumption on the part of those researchers - that software testing should be, for the most part, a systematic, standardised and automated process . If so, then the key abilities required for a software tester are to be able to use the techniques, as implemented in automated tools - beyond that, the only insight and creativity required is to determine which technique or tool is most appropriate for the testing task.

On the other hand, a relatively small number of practitioners and researchers have considered testing as a creative human activity. According to Kaner, Bach, and Petticord [7], manual and automated testing complement each other; automated testing cannot substitute for manual testing, as human variability and insight can reveal bugs that cannot be found automatically, while automated testing can "extend the reach" of the tester to make more comprehensive checking possible than would be feasible manually. According to Armour good testers have a nose for testing [8]. Their intuition drives them in testing. Iivonen et al. [9] found that highly performing software testers exhibit thoroughness conscientiousness, patience, persistence. Shoab et al. [10] found that extraverts were good at exploratory testing. Beer and Ramler [11] found that experienced software testers possessed increased domain knowledge that helped them in their job. Similarly, Itkonen et al. [12] concluded that experience is important in software testing. Capretz et al. [13; 14] suggested software testers should have thorough and acute attention to details and good organizational skills. Shah and Harrold [15] and Rooksby [16] found that testing is considered a boring job. Therefore software testers should possess the quality of dealing with boring and sometimes monotonous jobs. There is, overall, support in the literature for a view of software testing as a task critically dependent on the personal characteristics of the tester.

The limited available research reinforces the assumption that different human factors might have strong influence on the effectiveness of software testing. Most of these human characteristics are strongly connected with personality traits. Influence of those characteristics suggests that certain personality traits have similar influence on the performance of software testers.

To identify the personal characteristics, such as personality, that influence the performance of software testers, a validated and reliable instrument to assess software testers' performance is an essential prerequisite. However, from an extensive search of relevant literature we did not find any widely accepted and well established performance appraisal method for software testers. We found that Kaner [17] has advocated for qualitative assessment of bug reports and has discouraged to consider number of bugs found by software testers to assess their performance. These are useful criteria to assess software testers' performance against, however no empirical data is available on how these can be implemented to assess software testers in practice. This indicates a gap in research relating to performance of software testers.

From the above discussion it is evident that there is a lack of information on software testing unit job responsibilities. However, the goal of the role indicates an alternative approach is necessary for successful software testing. There is limited research available on the influence of different personal characteristics of software testers on their effectiveness. There is also a deficiency of an accepted instrument for software testers' performance assessment. All these encouraged us to design this research. The research is designed to enhance our knowledge on software testing unit job responsibilities, to find the human factors that influence software testers' performance and to find specific nature of such influence of personality traits on varying level of tester performance. We also aimed to find if the personality profiles of software testers differed significantly from other practitioners involved in software development. We also propose a novel Performance Appraisal Form (PAF) for software testers and attempt to refine the form based on feedback on the form collected from software development project managers.

1.4 Research Problem

Software testing refers to executing the software to verify if it meets specified requirements. The ultimate goal of software testing is to ensure reliability of the software. In achieving the goal software testers do not only rely on executing the software. Their role begins from analysing requirements even before the software is developed and continues with the software development. Along with finding bugs using efficient tools and technologies and reporting detected bugs in precise way, software testers also ensure the detected bugs are fixed. However, the diversity of the responsibilities of this role has not been analysed or documented that holds back further research on this profession.

From the limited empirical knowledge on the diversity of the role of software testers it is assumed that the person carrying out this role is critically important for the success in this role. The limited available research also indicates that certain human characteristics are influential to the success of this role. Most of the characteristics are related to personality. However, the effect of personality traits on the effectiveness of software testing is unknown. Also whether software testers possess certain personality traits that are distinct from others is unidentified.

To find the influence of human characteristics such as personality on the effectiveness of software testing, a means to distinguish among different levels of effectiveness is necessary. However, to our knowledge there is no well accepted standard instrument to assess effectiveness of software testers.

1.5 Research Significance

The research is designed to address the research issues outlined in Section 1.4. The research is divided in a survey on the factors affecting software testers' performance and influencing software test team development, a survey of collecting information on software testing job responsibilities, a survey of software testers' personality profiles, a survey of state of practice of performance appraisal of software testers, and a quasi- experimental study investigating the effect of personality traits on the effectiveness of software testing. As part of the research we also proposed and refined a Performance Appraisal Form (PAF) for software testers based on the feedback from software project managers.

The research is designed to investigate different aspects of software testing that should enhance our knowledge about this profession. The research is significant for both software testing research and practice.

1.5.1 Software Testing Research

We are unaware of any agreed upon list of responsibilities of software testers. Such a list is important to design research related to software testers. The refined list of software testing job responsibilities should help researchers defining the scope of the role and to draw the line between software testing and other closely related software development roles. This should also help the research finding who is best suited for the role.

To find the factors that influence software testers' performance a measure to assess the performance is needed. We found that Kaner [17] proposed detailed assessment approach for software testers'. However, the proposed approach of performance assessment may not be practical in research context.We believe, the information on how software testers' performance should be assessed and the proposed Performance Appraisal From (PAF) for software testers may help research in software testing requiring to evaluate testers.

1.5.2 Software Testing Practice

The refined list of software testing job responsibilities should help employers to plan and design roles for software testers. This should also give young IT graduates who want to take software testing as their career, an idea of the role.

If specific personality traits which have influence on the effectiveness of software testing can be formed, the traits to be looked for while recruiting software testers can be suggested. This should help the recruitment process of this profession which still relies on the experience and judgment of the recruiter. This should also help young IT graduates to select appropriate IT career for them.

The proposed Performance Appraisal Form (PAF) of software testers can be used in software development organization to assess software testers' performance for promotion, remuneration and other organizational purpose.

1.6 Research Contributions

This research has made several contributions in the discipline of software testing from research and industry perspective. Some of the contributions are described as follows:

- From the review of literature we identified several “gaps” in research relating to software testing. We designed research studies to address some of these gaps.

- We proposed refined list of software testing unit tasks will help young graduates to get better understanding of this role. This will also help recruiters to design job responsibilities for testers and researchers to design detailed research studies.

- Several factors were identified to be influential to the performance of software testers. The empirical evidence of the factors to be influential to the performance of software testers will encourage researchers to investigate these influences in more details.

- We found personality traits such as open mindedness, extraversion and conscientiousness are connected with performance of software testers. The personality traits found to be associated with the performance of software testers, will help designing research studies to find the specific effect of each personality traits and also will help young IT professionals in selecting their career.

- We proposed and refined a Performance Appraisal Form (PAF) for software. The form can be used in industry context for performance appraisal of software testers and in research context requiring assessment of testers' performance.

1.7 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 presents our review of literature related to personality and other human factors of software testers, personality of other software development practitioners and notes from industry and academic experts on software testers. This chapter also reports our review of relevant literature in performance evaluation of software testers and other software development practitioners.

Chapter 3 contains background knowledge on personality, personality traits, personality theories, personality assessment tests and approaches to performance appraisal of employees.

Chapter 4 gives an overall description of our research. The chapter details the formal research questions investigated through different research studies reported in subsequent chapters, the research methodologies adopted for different research studies and the instruments used.

Chapter 5 partially describes our preliminary multi part survey, The part of the survey reported in this chapter focuses on factors influencing individual software testers' performance and reports the results. The survey contained questionnaire on different factors such as automation tools and techniques, experience, characteristics, training/certification and performance of software testers. The participants gave valuable opinion on the influence of those factors on testing and listed some more factors that they thought important.

Chapter 6 describes the second part of the peliminary survey reported in last chapter. This part of the survey focuses on software testing team development.

We collected opinions of the participants on the influence of different factors, necessity of team diversity and the role of experience on software testing team development.

Chapter 7 describes a triangulated research study to find software testers' job responsibilities and reports the findings. In this research study we adopted different research methodologies to collect information on software testing unit tasks. The sources of information include worklog of software testers, job advertisements and bug reports collected from open bug repositories. We analysed the collected data from different sources to get idea of software testing unit job responsibilities.

Chapter 8 reports the results of a survey conducted to find personality traits of software testers. The personality profiles of software testers are compared to other software development practitioners and the significant differences are reported in this chapter.

Chapter 9 describes a quasi experiment conducted to find influence of personality traits on the effectiveness of software testing. The big five factor of personality is compared to different metrics of testing performance and the association found are reported in this chapter.

Chapter 10 presents the results of a survey of state of practice of performance appraisal of software testers. The survey also collected feedback on a proposed Performance Appraisal Form (PAF) for software testers. The proposed form was refined based on the obtained feedback as described in the chapter.

Chapter 11 lists and details the web based tools that are used for data collection for the surveys conducted as part of this thesis. This chapter contains the design details of all the web based tools we used.

Chapter 12 summarizes and discusses our finding of the series of research studies conducted as part of this thesis.

Chapter 13 concludes the thesis with future research plans.

Chapter 2

Related Work

2.1 Introduction

This chapter presents the findings of our review of research related to this thesis. This begins with a review of the role of software testers. We also review the published opinions of expert practitioners and academics on who is a good software teters. We present review of research relating to personality of software testers. We then review the research related to personality and human factors of programmers and the broader software engineering professionals. We then broaden our focus and include research on other human factors of software testers. Finally we present our review of research related to performance of software testers and programmers.

2.2 What Do Software Testers Do

The concept of software testing refers to the job of finding faults, checking the fitness and ensuring a certain level of reliability of software applications before deployment. Turing's article [18] published in 1950, was one of the first written notes on program testing that proposed to differentiate the program behaviour with a reference system by "an investigator" (a tester). This preliminary concept of testing is analogous to what we find in the classic book "The Art of Software Testing" where Myers [19] describes that testing is a process of executing the program with an intention to find faults.

The primary focus of software testing is to find faults. However, with the growth of software testing this has extended from finding faults to prevention of faults. Gelperin and Hetzel [20] have classified five different stages of testing growth according to the varying focus. Until 1956, testing was mostly debugging oriented. Baker differentiated testing from debugging [21]. From 1957 - '78, testing was more about demonstration that the program satisfies the specification. Two later periods of testing (1979-1982 and 1983-1987), called destruction and evaluation oriented periods respectively, were more focused on finding faults. The last and ongoing period of testing started from 1988 is referred to as prevention oriented where the main goal of testing is to prevent faults.

Testing now includes a number of other tasks along with running the program to find faults. Kaner [7] suggests that the software testers' role should include versatile tasks from ensuring some standard process of testing to being a customer advocate. The first testing maturity model, called TMM, was published by Illinois Institute of technology [22]. The model aligned with Beizer's [23] mental model of testers, mentions five different levels of maturity ranging from level 0 where testing and debugging are indistinguishable to level 4 where testing supports development of a low risk product. TMM levels are parallel to the capability maturity model of software development CMM [24]. Kasse [25] has described the job responsibilities of quality assurance practitioners according to the CMMI standard. Along with testing the list of responsibilities includes different jobs from assisting managers in developing and maintaining quality goals to serving as internal customer representative in front of development team. Although process standards and the software testing books mention different roles of software testers, a defined set of such roles is still missing.

Capretz and Ahmed [13; 14] collected the job responsibilities mentioned in different job advertisements for different software engineering roles including software testing and connected these responsibilities with required human skills. They suggested that software testers should have attention to details and possess good organizational skills. These are common in sensing and judging type people, as categorized by the MBTI personality assessment tool [26]. Capretz and Ahmed's [13; 14] approach was a good attempt to collect the unit job responsibilities of software testers. However, most of the advertised responsibilities are related to the custom needs of the job provider. Such collection for a portion of time may not guarantee all actually desired and practiced responsibilities of testers. This lack of clarity on software testing job responsibilities impedes research on finding who is best suited for the role. While managing the testing process and ensuring standards of testing may be more effectively performed by a person who is more disciplined, getting bugs fixed and preventive teaching to developers require good communication skills. As the testing community lacks an generally agreed upon set of activities required for the role, the question “who is a good software tester?” is to some extent unanswerable and as such, still remains unanswered.

2.3 What Makes A Good Tester - Expert Views

A number of experts have expressed their views on the desired characteristics of software testers. According to Armour [8], good testers have a “nose” for testing and they have an intuition that helps them to determine what and how to test. Pettichord [27] listed a number of distinct characteristics of programmers and software testers. According to him, software testers should tolerate tedium, be sceptical and be comfortable with conflicts, while programmers should automate tedium, be believers and avoid conflicts.

Pol [28] and Black [29] suggested some characteristics of software testers. While Pol suggests that software testers should be creative, accurate and strict in their methodical approach, Black thinks that software testers are “professional pessimists” who possess the curiosity of looking for faults. Weyuker et al. [24] have outlined general and technical skills required for a software tester based on the practices followed in AT&T. However, some of the skills such as good oral and written presentation skill, planning and design skills seem to be important for all who plan to take software testing as their career.

According to Burnstein [30], good communication skills, problem solving and team playing capability are important for software testers. The author also suggested software testers should be creative and open to new challenges.

2.4 Personality of Software Testers and Debuggers

Most software testing research, to date, has emphasized on the technical side of the discipline [6]. Research investigating human attributes such as specific personality traits of software testers is thin on the ground. Until recently, research investigating the influence of personality of software testers on their effectiveness was uncommon. However, we find a body of research into personality impacts with software engineers in general. The impact of personality on the area of programming has also gained little attention to date. Similar research with debugging or testing, however started much later.

In 2007, Da Cunha and Greathead [31] examined the connection of MBTI personality types of students with the ability in a debugging task. Due to sample size drawbacks they could not consider all four dichotomies of MBTI. They found that logical and ingenious people categorized by MBTI were good at code review task. While Da Cunha and Greathead used MBTI, Almodaimeegh et al [32] assessed the personality traits of programmers with locus of control. Locus of control is described as the extent to what one believes that an event is the result of external factors than his/her own effort [33]. They also investigated the influence of social learning style on debugging skill measured with code comprehension, bug detection and bug repair test. Their study revealed that although there is no significant relation between locus of control and debugging skill, individual social learning style and experience are influential on debugging performance. Although debugging can be viewed as a part of testing, testing includes many other tasks besides debugging. Thus the studies focusing on debugging do not give the full picture of software testing and personality impact.

Shoaib et al. [10], studied the effect of personality traits of students, assessed with MBTI, on the effectiveness of exploratory testing. They concluded that extrovert personality traits have a positive correlation with effective exploratory testing. Exploratory testing is a new concept in testing where the tester learns as the testing progresses and utilizes the learned experience while running tests. It is defined by James Bach [34] as *"simultaneous learning, test design and test execution"*. Learning expressed as experience is the core point of exploratory testing. However, this is a specialized testing technique and, based on the findings, while it can be predicted that extraverts can be good exploratory testers, whether extraverts will be good testers in general remains an open question.

Capretz and Ahmed [13; 14] collected required soft skills listed in job advertisements for IT practitioners and hypothesized what soft skill is required for what IT role. The authors suggested that sensing and judging people assessed with MBTI will be good at software testing based on the type of soft skills required for them. These two types of MBTI are equivalent [35] to openness to experience and conscientiousness of big five factor model. Rehman et al. [35] mapped the soft skills required for software testers with big five personality traits

Table 2.1: Studies Related to the Effect of Personality on Debugging and Testing

Study Identifier	Instrument used for personality assessment	Area	Traits having positive influence
[31]	MBTI	Debugging skill	Logical and ingenious
[32]	I-E scale and the learning style inventory	Debugging skill	Individual social learning style and experience
[10]	MBTI	Exploratory testing skill	Extraversion
[13; 14]	MBTI	Software Testers (The study also included other stakeholders)	Sensing and judging

and suggested that openness to experience and conscientiousness are important for the role of software testers. These suggestions were based on the experience and perception of the authors. No empirical data was provided.

The outcomes of the studies related to debugging and testing performance are summarized in the Table 2.1.

Most of the studies described in Table 2.1 employ MBTI for personality assessment. While MBTI is observed to be most commonly used in this kind of research, there are some debates about the reliability of the test [36; 37; 38]. The test suffers from the lack of any negative judgment and each of the types is described in an acceptable and glib manner. The descriptions are more general and any description can fit with a wide range of people. This lack of specialized assessment of the test is known as the Barnum effect [36]. Many other limitations of the test are described in [37]. Bjork and Druckman [38] have questioned most of the studies that has used MBTI. The authors also suggest investigating the utility of MBTI in an organizational setting more than has currently been done. The above studies illustrate that the MBTI test is not widely accepted among the psychologists' community. Thus, the limited literature on the effect of personality on testing and debugging ability and the use of a controversial instrument in those limited studies found in the literature, do not help us to predict who can be an effective software tester which demonstrates potential for useful insights investigating the issue.

2.5 Personality of Programmers

There is a body of research associating personality types with the effectiveness of programming. Until early 2000, most of the research of this category investigated whether a specific personality trait measured with MBTI was over represented in programming community. The studies of Sitton and Chmelir [36], Bush and Schkade [12], Lyons [39] and Chandler et al. [40] all are examples of this kind of research. Later the research included the impact of the personality types on performance.

Cegielski and Hall [41] explored the predictive power of personality along with theoretical value belief and cognitive ability on the performance in object oriented programming. Personality in this study was measured with self esteem, self efficacy, locus of control and neuroticism. They found that personality type was a predictor of performance in object oriented programming. Darcy and Ma [42] used five factor model of personality to find the influence of personality on the performance of students as programmers. The participants of the study were given a programming task. They concluded that there was no significant difference in personality between the group that completed the task and the group that did not complete the task.

There has been some personality related research undertaken on code comprehension and software design. Arochiam et al. [43] investigated the connection between gender and personality traits measured with Rajan's 12 point questionnaire and object oriented code comprehension skill measured with a standard questionnaire in C++. They found that male student participants with cooperativeness and high emotional ability did well in code comprehension. Their findings cannot be directly compared with Greathead's [44] observation that introvert people are good at code comprehension, as the later study employed MBTI personality types for personality assessment. Ahmed et al. [45] studied the connection of MBTI personality types of students with their performance in software design course. They found that judging and thinking type students did better in this course.

2.6 Personality of Software Engineers

There is also some research analysing the personality traits of software engineers in general. Capretz [46] administered the MBTI personality assessment test on 100 software engineers ranging from student to professional level. They found that ST, TJ and NT types were over represented in the sample. The author also suggested that ET type would be better for system analyst, INTP for programming and E/ISTJ for managing, and TP for technical. In his further studies with Ahmed [13; 14], Capretz associated different required soft skills of software engineers collected from job advertisements with MBTI personality types and suggested different personality types suitable for different roles of software engineering.

Sach et al. [47] recently analysed five research studies conducted from 1985 to 2010 using MBTI to find the most common personality types of software engineers. All the studies reported similar findings. The combined results suggested that certain MBTI types are overrepresented among software engineers. Those types are different to the general population of the United States. More specifically, thinking and judging types assessed with MBTI were over represented among software engineers. A systematic literature review on personality research with software engineers conducted by Cruz et al. [48], summarizes the body of research on personality of software engineers using different personality assessment tests. They reviewed 42 research studies and found the majority of personality research was done in pair programming followed by team effectiveness. The MBTI test was used in most of the reviewed research. The authors suggested that collaborative research work between software engineering and psychology researchers is required on this topic.

Sodiya et al. [49] prepared a general test that assesses the five personality factors along with cognitive style with a standard questionnaire and suggests the software engineering role that best suits the personality. Feldt et al. [50] studied the effect of personality of 47 professional software engineers, measured with 50 item IPIP inventory on the attitudes of software engineers and found that there are different clusters of personalities among them and each cluster have significant correlation with attitude. Clark et al. [51] assessed the personality of a number of IT students and professionals with Adjective Check List (ACL) and found that both exceptional students and professionals are conscientious, however, exceptional students are introvert while exceptional IT developers are extravert. The difference in type between exceptional students and developers raises an issue that exceptional students might not be exceptional developers later.

The diversity of personality in software engineering teams has also been investigated in several studies. In their series of ethnographic studies Karn and Cowling [52; 53] observed the team interactions of small software engineering teams of students and assessed the personality of each team members with MBTI personality assessment test. They found that heterogeneous teams (i.e. teams having diverse personalities among their members) showed better performance. This is similar to the findings of Pieterse et al. [54] that states personality diversity and team ability have positive impact on team performance. They also mentioned that team diversity helped the team to become productive sooner. The higher productivity of heterogeneous teams was also supported by the study of Rutherfoord [55]. However, the claim that diverse teams have a positive influence on team performance, contradicts with the study conducted by Gorla and Lam [56] on 90 IS professionals from 20 teams. They found some significant relations of different roles among the team with specific personality traits measured with Keirsey temperament sorter. They also concluded that there is no significant influence of diversity on the team performance. Peslak [57] conducted a pilot study and concluded that higher level of extraversion, thinking, and judging personality type are positively related with team success.

Misra et al. [58] investigated the effect of 14 factors on the success of agile software development practices with a survey. Personal characteristics of the agile team members were one of those. They found that personal characteristics have a positive association (although not strong) on the success of agile software development practices. In the open ended responses to their survey they found that a number of personal characteristics were considered important by the respondents. Those include: having intellect, taking responsibility, self criticism, being passionate, and not being perfectionists.

From the above discussion it is evident that there is limited research on the effect of personality on testing in comparison with research exploring influence of personality on performance in programming, code comprehension and different roles in software engineering. There is enough evidence that personality of a tester might be an important factor in successful software testing. The limited research in this field encourages researchers to explore this area.

2.7 Other Human Factors of Software Testers

Software testing is a creative human activity. The human attributes that influence the effectiveness of a software tester is highly important to be investigated for the improvement of this discipline. Apart from personality, the influence of other attributes on the effectiveness of software testing that has been investigated includes experience and organizational issues.

Beer and Ramler [11] explored the effect of experience on the effectiveness of software testing. In their field based study they found that experienced testers used their domain knowledge to “fill in the gaps” in incomplete and ambiguous specifications. On the other hand Juha et al. [12] studied 11 software testers to answer the question “How do testers do it?”. They found that software testers use a number of techniques and strategies for testing and they do not always rely on documents. Test execution techniques are strongly based on experience of the software tester and tests are run in non-systematic fashion. These studies emphasize the importance of experience in software testing.

Organizational issues in software testing have so far also attracted little attention. In their case study in a company at three different locations, Shah and Harrold [15] found that testing is considered to be a boring job and that junior and senior testers have different motivations and attitudes towards software testing. This finding complements the outcome of an ethnographic study conducted by Rooksby et al. [16], where they found that testing is a boring task. They found that most of the problems related to testing cannot be dealt with technical solutions only. Sometimes cooperative practices help to solve these problems. They also found that software testing problems are mostly organizational and those should be attempted to solve from organizational or inter organizational means [59]. Cohen et al. [60] discussed different conflicts in software testing from organizational and employee’s point of view.

Capretz et al. [61] analysed the soft skills listed in different software engineering job advertisements published across North America, Europe, Asia and Australia. They found that for software testers having good communication skills was highly required. Good analytical and problem solving skills and organization skills were moderately sought after in the studied job advertisements. The authors concluded that employers are not giving much importance to all the important soft skills needed for software testers.

Over all, there is limited evidence of research on different human attributes in software testing. The few studies mentioned above explain the attributes of a software tester such as coping with monotonous job and cooperativeness. These characteristics of human are related to certain personality traits. The people having relevant personality traits will exhibit the characteristics and hence will be suited for the role. The investigation of such characteristics and relevant personality traits are, therefore, of considerable importance.

2.8 Performance Assessment of Software Testers

To determine the effectiveness of testers some form of performance appraisal of software testers is necessary. However, not much published research is available regarding performance evaluation of software testers. Fenton and Pfleeger [62] suggest to measure efficiency of software testing with the number of bugs found per KLOC. Grady and Caswell [63] suggest looking for average reported bugs per working day. Kaner [64] has discussed some of the risky impacts of taking into account only bug count to measure software testers' efficiency. According to Kaner, bug counts would be influenced by other factors like reliability of code being tested, difficulty of testing the code, testing techniques being used (for example, exploratory and regression testing will produce different bugs).

In a different article, Kaner [17] proposed a multidimensional assessment method for software testers. He emphasized qualitative assessment. His recommended approach suggests evaluating a software tester for a long time, preferably on a weekly basis. He suggests taking testers' plan of testing, execution of tests and bug reports in to account for evaluation. He suggests the reviewer conducts small discussions with the software testers regarding their test progress to obtain information. Kaner's proposed approach seems to be an effective way of evaluating testers. However, the effectiveness of this approach is not supported by any research results. Also, the evaluation is largely dependent on the perception of the reviewer and such long term evaluation methods might appear to be infeasible for a number of organizations. The approach is time consuming and required highly trained managers to successfully conduct the appraisal without any bias.

2.9 Performance Assessment of Other IT Practitioners

In this section we present our review of some approaches that are used for evaluation the performance of IT professionals. Killingsworth et al. [65] described a model to motivate and evaluate information systems staffs. The model is used by a number of large organizations. In this model an employee is evaluated on the following five factors:

- Product Quality: Employee's contribution to the delivered product or service quality.

- Customer Outreach: Employee's contribution to maintaining and expanding work with current client and to wining new client.
- Staff Development: Employee's contribution to enhance individual and team effectiveness.
- Administrative Efficiency: Employee's contribution to develop administrative procedures as well as maintaining the procedures with punctuality and accuracy.
- Fiscal responsibility: Employee's own financial plan. Employee is also evaluated against organization's financial plan for the employee.

Senior project manager and team leader assess each employee against each of the five factors. A five point rating scale is used for the assessment. Each factor accounts for varying weights for the review of the senior manager and the team leader. For example product quality accounts for 40% of the review for a team leader and 20% of the review for a senior manager.

Mayer and Stalnaker [66] describe a number of methods that are useful for selection and evaluation of computer personnel. While most of the methods presented in their paper are useful for the selection of programmers, very few of those can be used for evaluation of programmers as well. The authors describe Dickmann's [67] Programmer Appraisal Instrument (PAI). In this method programmers are reviewed by the supervisors on four major areas of performance-professional preparation and activity, programmer competence, dealing with people and adapting to the job. There are a total of 42 questions to assess performance on those areas. Supervisors rate the programmers with the help of a five point scale.

42 items are also used in Bairdain's approach [68]. This approach considers the following factors: programming knowledge/capability, working style, temperament traits and personal professional items. The ten highly important items of this approach are related to performance in programming where as the ten least important items are related to personal-professional attributes.

The last evaluation method described by Mayer and Stalnker is Berger and Wilson's [69] Basic Programmer Knowledge Test (BPKT). The test is not specific to any programming language. The test evaluates the knowledge of the programmer on the six areas- logic estimation and analysis, flow diagramming, programming constraints, coding operations, program testing and checking and documentation. The knowledge is assessed with 100 multiple choice questions.

Powell [70] presented 13 categories with definition for each of the category to rate programmers and analysts. Rating is conducted with a five point scale. Powell has proposed a distribution of performance according to his method for a group of 20 programmers and analysts.

2.10 Summary

Software testing includes a number of tasks, such as executing and reviewing software, communicating found faults to others, participating in requirements analysis and so on. However, a comprehensive and widely agreed list of these activities is currently missing. This gap impedes the research investigating characteristics of effective software testers.

The limited literature and the conceptions of the expert professionals suggest that effective software testers have some specific characteristics. However, such characteristics have not been linked to personality traits and the desired personality traits for the effectiveness in this role has not been established by research so far. On the other hand, such research in other, related disciplines, such as programming, have a long history.

There is no doubt that the advancement of new techniques and tools in software testing has helped to enrich the discipline. There is enough evidence that human factors also have some positive influence on the effectiveness of software testing. Thus, this kind of research also seeks some attention by the software testing research community.

To find the characteristics of high performing software testers, we need to recognize them. Unfortunately a well accepted measure of the performance of software testers is unavailable. This indicates a gap in the research and practice of software testing.

The possible benefits of such research into software tester tasks, performance appraisal and personality impact on testing effectiveness could include: better understanding of software testing practices; better recruitment of testers; better development and management of testers; and improved performance overall by testing and development teams.

Chapter 3

Background

3.1 Introduction

Assessment of personality and software testing performance plays an important role in this research. As such, a knowledge of research and practice in these areas is necessary to understand and appreciate the motivation and direction of this research. This chapter provides a foundational knowledge about theories and practice related to human personality and performance appraisal.

3.2 Personality

Personality is one of the most used words to refer to a person. However, defining personality is a difficult task. The struggle for a useful definition began with the authors of first text book on personality, Gordon Allport and Henry Murray [71].

The word personality came from the Latin word persona via French. Persona means a mask that is worn by an actor to portray a particular character. There have been a lot of approaches to describe personality. Some have said it's the personhood or the individuality of a person [72], whereas some have thought personality is personal charm [73]. According to Haslam [73] personality encompasses the sort of non intellectual psychological characteristics that are most informative about an individual and that help to describe the differences between people. It is also a feature of an individual that is organized and relatively enduring and that influences the person's interactions with others and their adaptation to their social environment.

3.2.1 Personality Traits

The criteria by which people differ from each other are called personality traits. Conley defined personality traits as- *"personality traits constitute very generalized behaviour patters in response to emotional tendencies"* [74]. Traits are representative factors that are predictive of one's behaviour patterns, feeling, thinking and related activities.

3.2.2 Personality Theories

There are a number of theories that have been developed to describe personality. The Freudian theory states that personalities differ in the amount of psychic energy and the method of handling it [75]. Some other theories like "trait and type theory" are developed based on separate characteristics of a person [75].

There have been a few theories developed to help in choosing a set of words that represent personality traits. The first attempt was taken by Allport and Odbert [73], who took an English-language dictionary containing 550000 words, and identified about 18000 words defining traits. They refined those words and ended up having 4500 words. In order to get a smaller list than 4500 trait terms as listed by Allport and Odbert, Raymond Cattell [76] sorted the words into cluster of synonyms and near synonyms according to his personal judgement. He identified 160 such clusters. He used the following two statistical methods to group similar traits:

1. Correlation coefficient: Correlation coefficient indicates the association between two variables. It can vary from -1 to +1 [73].
2. Factor analysis: Factor analysis determines patterns within a group of correlation that describes a factor [73].

He found 12 factors after applying correlation coefficient and factor analysis on the clusters. He added four more terms from his additional studies. Finally he got 16 personality factors known as 16PF [73].

Using similar techniques as the above, other psychologists derived *big five factors* for analysis of personality [74]. These are described below:

1. Extraversion (E) Carl Jung, proposed that person's libido can flow in one of two directions-outward and inward. He substituted the concepts of psychic energy and motivation for libido. The dimension he introduced is Extraversion - Introversion. He concluded that humans can fall into only

two categories - extravert or introvert. Later Hans Eysneck introduced a middle range called ambivert [73].

2. Agreeableness (A) Agreeableness encompasses the expressive quality of admirable human aspects of personality. People who are high on agreeableness are helpful, sympathetic, patient, cordial, cooperative. People who are on the other end of the continuum, however can be cruel and untrustworthy.

3. Conscientiousness (C) Conscientiousness incorporates qualities like hard working, discipline, dutifulness, well organization, and so on. High conscientious people are efficient, organized and dependable. They accomplish tasks in systematic ways. Low conscientious people, on the other hand are inefficient, careless and haphazard.

4. Neuroticism (N) This factor covers all the forms of excessive emotionality. Watson and Clark [77] suggested that this personality dimension is concerned with negative emotionality of people such as anxiety, depression, angry hostility.

5. Openness to Experience (O) Openness to Experience is associated with intelligence and intellectual interests. Many Scientists have named it differently. Such as “inquiring intellect” by Fiske [78] and “intellectance” by Digman [79].

There are also some popular three factor models. According to Hans Eysenck [80] the three main factors are extraversion-introversion, neuroticism and psychotism. The first two factors are already described in the big five factor model. The third one, psychotism, he defined as a factor that is composed of traits like aggressiveness, coldness and other such anti-social tendencies. Tellegen [81] named the three factors as positive emotionality, negative emotionality and constraint. The first two are closely related to the first two factors of the Eysenck model. However, the last one has a strong negative association with the third factor of the Eysenck model. Watson and Clark developed another three factors. These are Positive temperament, Negative temperament and Disinhibition [74]. Disinhibition is the opposing pole of constraint. Although the names of each of the three factors may vary, they converge towards three main distinct domains of personality.

3.2.3 Personality Assessment

There is a number of ways in which personality data can be obtained [82], including self reported data, observed data, life time data and test data. Personality research is conducted in two main ways, the correlational approach and the experimental approach [83]. A number of concepts of personality assessment using both the approaches are available. There are also a number of tests to identify the personality traits of an individual [73]. Each of those has unique characteristics and is used for specific purposes.

A number of techniques have been developed for personality assessment. There are two popular approaches for personality assessment, the clinical approach and the statistical approach.

3.2.3.1 Clinical Approach

The clinical approach is used for projective personality tests. Projective group of tests consider responses of an individual to ambiguous stimuli. Projective tests are mostly used for assessing psychological disorders. These tests are time-consuming and the data obtained by the test are analyzed using a clinical approach. Some popular projective tests include Rorscharch method, Thematic Apperception Test (TAT) and Edwards Personal Preference Schedule (EPPS).

3.2.3.2 Statistical Approach

Statistical approaches like correlational and factor analysis methods are widely used to come to a type indication for personality assessment. Most of the statistical tests are designed on the popular personality models discussed in previous section. MBTI, NEO PI-R and such personality tests are generally used for self-assessment or organizational purposes. A brief discussion of some of these tests is given below:

Myers-Briggs Type Indicator (MBTI) Carl Jung, a follower of Freud, proposed typological theory in his 1921 book “Psychological types” [84]. According to the theory, there are two dichotomous pairs of cognitive functions: “rational (judging)” and “irrational (perceiving)”. He suggested these functions are expressed in either an introverted or extraverted form.

Isabel Briggs Myers and her mother Katharine Briggs developed the Myers Briggs Type Indicator (MBTI) based on Jung’s theory. Table 3.1 illustrates the concept of the test: [26]

Table 3.1: MBTI Description

Category	Type Code	Definition
Mental Processes or function	Sensing(S) - Intuition(N)	How we perceive-gather and take information
	Thinking (T) - Feeling(F)	How we evaluate choices and reach conclusion
Mental Orientations or attitude	Extraversion (E) - Introversion (I)	How we get and use our energy
	Judging (J) - Perceiving (P)	How we organize and plan

The permutation of the four preference dichotomies results in sixteen personality types that form the MBTI inventory. Any of the types result in four character patterns which show a hierarchy. In order of influence and importance the elements of the hierarchy are termed as “Dominant” –> Auxiliary” –> “Tertiary” –> “Inferior”.

Sensing (S) and Intuition (N) are two contrasting ways of taking in information and are called Perceiving Functions. Similarly Thinking (T) and Feeling (F) are two contrasting ways of making decisions and hence are called Judging Functions.

MBTI is the most popular personality type indicator at the present time. For most cases, 75% of the time it gives same result if a person retakes the test. In the case of clear preferences, it gives the same result 95% of the time [85].

MBTI is a simple and inexpensive personality test. Another reason behind MBTI’s popularity is the very good promotion of the test. The publishers have done a great job in promoting this test and it also has some intuitive appeal [86].

The test has been criticized for lack of negative judgment referred to as Barnum effect [36]. However, this may be important for acceptance for use of the

Table 3.2: SFPQ factors and facets

Factor	Facets
Agreeableness	Abasement
	Even-tempered
	Good-natured
Independence	Autonomy
	Individualism
	Self-Reliance
Methodicalness	Cognitive Structure
	Deliberateness
	Order
Extraversion	Affiliation
	Dominance
	Exhibition
Industriousness	Achievement
	Seriousness
	Endurance
Openness to Experience	Change
	Understanding
	Breadth of Interest

results in the workplace.

MBTI inventory is distributed by CPP (formerly Consulting Psychologist Press) and it is a registered trademark of Myers-Briggs Type Indicator Trust.

The Six Factor Personality Questionnaire (SFPQ) The six factor personality questionnaire has a modified scale for Conscientiousness than the Big five factors [87]. It does not include Neuroticism. The test measures three facets for each of the six factors. The factors are described in Table 3.2.

Sixteen Point Factor Personality Questionnaire (16PF)

This questionnaire was developed to assess the sixteen factors of personality mentioned by Raymond Cattell [76]. The questionnaire includes two sets of forms. The first set contains forms A and B having 187 items each and the other set contains form C and D which are shorter. There is also a form named E, which is

Table 3.3: 16PF factors

No.	Factor
1	Warmth
2	Reasoning
3	Emotional Stability
4	Dominance
5	Liveliness
6	Rule-Consciousness
7	Social Boldness
8	Sensitivity
9	Vigilance
10	Abstractedness
11	Privateness
12	Apprehensiveness
13	Openness to Change
14	Self-Reliance
15	Perfectionism
16	Tension

specially designed for people with low reading skill. The sixteen factors are given in Table 3.3

NEO Five Factor Inventory (NEO FFI)

This test was designed to assess the big five factors of personality. The first version of this test was published by Costa and McCrae in 1985 [88]. They considered only three factors at that time, Neuroticism, Extraversion and Openness to Experience. Hence the Name “NEO” was given. This test examines six facets for each of the three domains. This personality inventory contains different versions of the test discussed in the following subsections. The descriptions are based on [89].

NEO Personality Inventory Revised (NEO PI R)

This test was published in the year 1992. This test measures all of the five domains of personality and six facets for each of them. A description of the facets and domains (taken from NEO PI-R manual [88]) are given in Table 3.4.

Table 3.4: NEO PI-R Domain and Facets [88]

Domain	Facet	High Scorer	Low Scorer
Neuroticism	Anxiety N1	apprehensive, fearful, prone to worry, nervous, tense, jittery, free floating anxiety	calm, relaxed, do not dwell on things that might go wrong
	Angry Hostility N2	frustration, bitterness	easy going, slow to anger.
	Depression N3	prone to feelings of guilt, sadness, hopelessness, loneliness, easily discouraged, often dejected	rarely experience such emotion
	Self-Consciousness N4	uncomfortable around others, sensitive to ridicule, prone to feelings of inferiority, akin to shyness and social anxiety	simply less disturbed by awkward social situations
	Impulsiveness N5	can not resist strong desire but often may regret later	easily resist such temptations
	Vulnerability N6	feel unable to cope with stress, becoming dependent, hopeless or panicked when facing emergency situations	capable of handling themselves in difficult situation

Extraversion	Warmth E1	affectionate, friendly, genuinely like people, easily form close attachments to others	more formal, distant in manner, reserved
	Gregariousness E2	enjoy the company of others	do not seek/ even actively avoid social simulations
	Assertiveness E3	dominant, forceful, socially ascendant, speak without hesitation, often become group leaders	prefer to live in the background, let others do the talking
	Activity E4	rapid tempo, vigorous movement, in a sense of energy, in a need to keep busy, lead fast paced lives	more leisurely, relaxed in tempo, are not necessarily sluggish or lazy
	Excitement seeking E5	crave excitement and stimulation, like bright colors and noisy environments, akin to some aspects of sensation seeking	feel little need for trills, prefer comparatively boring life
	Positive Emotions E6	laugh easily and often, cheerful, optimistic	merely less exuberant and high spirited
Openness	Fantasy O1	have a vivid imagination, active fantasy life, daydream not to escape but to make a inner world	Prosaic, prefer to keep their mind on the task at hand

	Aesthetics O2	have a deep appreciation for art and beauty, need not have artistic talent, not necessarily good taste	relatively intensive to and uninterested in art and beauty
	Feelings O3	experience deeper and more differentiated emotional states and feel both happiness and unhappiness more intensively than others	have somewhat blunted affects, do not believe feeling states are of much importance
	Actions O4	prefer novelty and variety to familiarity and routine, different hobbies over time	find change difficult, prefer to stick with the tried and true
	Ideas O5	enjoy philosophical arguments and brain teasers, does not necessarily imply high intelligence	have limited curiosity, if highly intelligent-narrowly focus their resources on limited topics
	Values O6	readiness to re-examine social, political, religious views	generally conservative
Agree-ableness	Trust A1	have a disposition to believe that others are honest and well intentioned	tend to be cynical and sceptical to assume that others may be dishonest or dangerous

	Straight-forwardness A2	frank, sincere, ingenuous	more willing to manipulate others though flattery, craftiness or deception
	Altruism A3	generosity, consideration of others, willingness to assist others in need of help	Self-centered, reluctant to get involved in the problems of others
	Compliance A4	tends to defer to others, to inhibit aggression, to forgive and forget, meek and mild	Aggressive. Prefer to compete than cooperate, express anger when necessary
	Modesty A5	humble, self-effacing	believe they are superior, may be considered conceited or arrogant by others
	Tender-Mindedness A6	moved by others' needs and emphasize the human side of social policies	more hard-headed, less moved by appeals to pity, consider themselves realistic who make rational decisions based on cold logic
Conscient-iousness	Competence C1	feel well prepared to deal with life	have a lower opinion of their abilities and admit that they are often unprepared and inept
	Order C2	neat, tidy well organized, keep things in their proper places	unable to get organized and describe themselves as unmethodical

	Dutifulness C3	adhere strictly to their ethical principles, scrupulously fulfill their moral obligations	more casual about such matters, may be somewhat undependable and unreliable
	Achievement Striving C4	have high aspirations levels, work hard to achieve their goals, diligent purposeful, have a sense of direction in life	lackadaisical, even lazy, not driven to succeed, lack of ambition, may seem aimless
	Self Discipline C5	have the ability to motivate themselves to get the job done	procrastinate in beginning chores, easily discouraged, eager to quit
	Deliberations C6	cautious, deliberate	hasty, often speak or act without considering the consequences

The test contains 8 questions for each of the facets. It can assess the facet scores as well as a combined domain level score based on the answers to the total of 240 questions. 10 questions from the original NEO PI were replaced in this version. There is no true/false type of answer in this test. Instead, it employs a five point Likert scale. One can respond either of Strong Disagree (SD), Disagree (D), Neutral (N), Agree (A) and Strong Agree (SA). The responses are scored as 4, 3, 2, 1 and 0 when SD is the direction of scoring. The scoring sequence is reversed when SA is the direction of scoring. So the total score on a domain can range from 0 to 192. However, for normal adults the range is defined as 25 to 172. It has three validity check question at the end of the test.

Scoring of the Test

This is a self scoring test. There are two ways of scoring: computer scoring and hand scoring. If the test is taken through a computer it will automatically score the test. The hand response can also be entered in the computer. The responses are summed to have a facet level as well as a domain level assessment. The score is then plotted in the given norm sheet to obtain the T-scores. There are two separate categories of profile form- one is college (17-21 years) and the other one is adults (21+). To fall into the "college" category one does not necessarily need to be a student attending a college.

Administration

This is a self-scoring test. The administrator should score the test together with the participant. Also the aim and process of using the test should be well described to the participant. NEO PI R is one of the very few tests that allow the administrator to read the questions to the participants in case if s/he does not understand it. However, this is discouraged to avoid any bias.

There are some response-related issues such as "no answer", "random answer" regarding psychometric tests. In the following subsections we discuss how NEO PI-R handles these issues.

No Answer

If a participant does not give an answer for fewer than 41 items, N (neutral) can be selected as those answers. However, individual facets should not be scored if more than three items are not selected. The test should not be scored if more than 16.7% items are omitted.

The administrator can query why answers were not selected. They can also remind the participant about choosing N if no suitable answer is found. However this is strictly discouraged since participants should try to answer each of the items as much as possible. Though NEO PI R allows selecting N for up to 40 omitted items, responses to those items will have a good impact on the overall result. Thus, there is a question about the validly of the result in that case. If a person does not have any association with some items at all, selecting N for those may not evaluate him or her perfectly on the related domain or facet.

Accurate answer

Validity check C asks the participant if all items have been endorsed at the right position. An answer of "NO" to this question actually invalidates the test.

Random Response

According to McCrae and Costa, one way of finding random responses is to note the same response for a number of consecutive items: 6 consecutive SD, 9 consecutive D, 10 consecutive N, 14 consecutive A and 9 consecutive SA.

Bias

Validity check A asks if items have been endorsed honestly. Again a response of D or SD to this invalidates the scoring of the test. Costa and McCrae suggested that if more than 150 number of responses are A or SA, than the participant might have acquiescence bias. On the other hand in case of less than 50 number of A or SA response, there is a possibility of having nay saying bias. In both the cases the test should be evaluated with caution.

NEO Personality Inventory 3

The original version of NEO PI R was used with 2000 adolescents in the year 2002. They did not answer 30 of the items due to having trouble in understanding these or not having attachment with them. There were 18 items that had an item total score on the facet scale less than 0.30. The 48 items had to be changed. However the publishers found a replacement for 37 items. The remaining 11 were taken from the original one. The resulting modified test was named NEO PI 3 and is the latest test from the NEO inventory of tests.

International Personality Item Pool (IPIP)

The tests discussed so far are all proprietary, as such the application of the tests are restricted by licensing agreements. Unsatisfied with this fact and the pace of development of the science of personality assessment, Goldberg [90], proposed personality assessments items and scoring mechanism named International Personality Item Pool (IPIP). This is a web based instrument that can be used freely. This test is popularly used in recent times [91].

3.3 Performance Appraisal

Performance evaluation, also referred as performance appraisal in an organizational environment, is the process of examining one's work behaviour and outcome against a set of criteria and providing him/her with the feedback explaining the scope of improvements. Performance evaluation is an integral part of management and is periodically conducted in organizations to make decisions about employee position, remuneration, promotion and supervision.

3.3.1 Approach to Performance Appraisal

There are different approaches to performance appraisal. The different approaches are briefly described here. Most organizations use all approaches in their performance appraisal system.

Performer Focused Appraisal This approach takes into account certain traits of the performer. The activity of the performer and the results of the activity are not paid much attention to [92]. This approach only discovers whether some qualities are exhibited by the performer or not.

Behaviour Based Appraisal This approach judges the performance on the work behaviour of the performer [92]. Two popular scales using this approach are described below:

1. Behaviourally Anchored Rating Scale (BARS): This scale has defined behaviour for each rating label. This is designed to assess performance in specific job.

2. Behaviour Observation Scale (BOS): This scale suggests that the appraiser observes work behaviour of the employee [92].

Result Focused Appraisal This approach includes assessment of performance based on predefined goals and objectives [92].

3.3.2 Scale Types

There are four main types of scale for assigning ratings to performance dimensions [92]. Different types of scales are described below:

1. Numerical ratings: This type of scale contains numerical rating labels, for instance 1,2,35.

2. Behaviour frequency: This type of scale considers the frequency of certain work behaviours. Rating label of this type of scale could include terms such as always, often, usually, seldom, rarely and so on.

3. Evaluation Concept: The appraiser makes a judgment on quality of performance and assigns a rating to the quality. The choices could include terms such as outstanding, competent, superior, satisfactory, unsatisfactory, marginal.

4. Compare against a standard: The performance is compared against a standard and a rating label is assigned accordingly. The ratings labels can be: exceeds the requirement, meets requirement, partially meets requirement, fails to meet requirement,.

3.3.3 Scale Labels

We can select three, four or five scale labels (choices) for rating [92]. The pros and cons of different number of labels are described in Table 10.2.8.

Table 3.5: Different number of performance lables [92]

Number of label	Advantages	Disadvantages

Three	1. Extreme performances are easily identifiable 2. Good for pass-fail assessment	1. Rare use of lowest level 2. No distinction between “need to improve” and “termination” condition 3. May not allow fine discrimination
Four	1. No midpoint 2. Sufficiently fine discrimination	1. No way of rating average performers 2. Rare use of lowest rate 3. May not distinguish between “need to improve” and “termination” condition

Five	1. More consistent with bell curve 2. Most managers are comfortable with it 3. Highest degree of familiarity and acceptability 4. True outstanding performers are judged	1. Appraiser may not have specific differences between levels 2. Rare use of lowest level 3. Increased central tendency

3.3.4 Appraiser (Who will appraise the performance?)

The following people can assess performance [92]:

1. Supervisor/Manager only
2. Peer (with supervisor/manager rating)
3. Customers (with supervisor/manager rating)
4. Subordinate (with supervisor/manager rating)
5. Self (with supervisor/manager rating)

A comparative analysis of different appraisers [92] is given in Table 7.2

Table 3.6: Pros and cons of different appraisers

Appraiser	Advantages	Disadvantages
Supervisor (manager) only	1. Most information about quality and quantity of work 2. Greatest stake in producing accurate appraisal of performance 3. Using one sources keeps the system simple	1. May be unduly and unwittingly influenced by personal relation
Peers (with supervisor/-manager rating)	1. Adds broader perspective. 2. Reduces political biases if large number of peers are included 3. Increased teamwork based strengths	1. Competitive co-workers may lower the rate 2. Increased paperwork and complexity 3. Longer execution time 4. Issues about confidentiality and privacy

Customers (with supervisor/manager rating)	1. Might use most important data	1. Might not be skilled enough for assessment 2. Increased paperwork and complexity 3. Longer execution time
Subordinates (with supervisor/manager rating)	1. Might use information from neglected source 2. Might obtain data unavailable from any other source 3. May reveal important information about supervision	1. May be reluctant to provide accurate rating 2. Increased paperwork and complexity 3. Longer execution time
Self (with supervisor/-manager rating)	1. Probably best source of data 2. Areas where supervisor and the employee disagree are highlighted	1. Probability of deliberately rating low or high

Correlation among different ratings Table 3.7 describes some studies that investigated whether employee ratings performed by different types of appraisers are correlated.

Table 3.7: Correlation among different ratings

Study	Type of correlation investigated	description of method	Results
Harris and Schaubroeck [93]	Self-supervisor, self-peer and peer-supervisor	1. Meta analysis 2. Searched for studies examining such correlations. 3. Found 36 self supervisor correlations, 23 peer supervisor correlations and 11 self peer correlations.	Correlation between self and supervisor ratings- 0.35, Correlation between self and peer ratings - 0.36, Correlation between peer and supervisor ratings - 0.62

Mabe and West [94]	Self rating and other means of ability assessment (test, grades, peer, supervisor ratings)	1. Meta analysis 2. Found 55 studies analysing self rating with other means of assessment of ability	Correlation between self and other means of assessment of ability - 0.29
William and Seiler [95]	Self - supervisor	1. Obtained self and supervisor rating from 202 engineers 2. Compared the ratings	Correlation between self and supervisor rating - 0.60

Pym and Auld [96]	Self - supervisor	1. Considered different groups of people- scientist, commercial apprentice, mechinists, technicians and programmers 2. Obtained self and different other ratings for different groups	Correlation between self and supervisor rating - 0.56
Klimoski and London [97]	Self - supervisor	1. Obtained self, peer and supervisor rating of hospital nurses using same questions	Correlation between self and supervisor rating - 0.05

The above studies reveal that there are low correlations between self and supervisor ratings. Thornton [98] suggests that the reason behind the low correlation is that individuals have a different view of their performance to their supervisor. However, the study of Harris and Schaubroeck [93] shows that supervisor and peer ratings have higher correlations than self and any other ratings.

Table 3.8: Different rating errors

Error	Definition
Halo effect	Rating person on one aspect depending on other aspect.
Central tendency	Rating people on the middle of the scale.
Negative and positive skew	Rating people lower or higher than actual performance.
Recency effect	Rating on minor incidents that have occurred recently than major one happened in the past.
Contrast effect	Comparing individuals with others while rating.
First impression effect	Rating on initial positive or negative judgment of the individual.
Similar-to me effect	Rating individuals higher who resemble the appraiser.
Attribution bias	Considering the factors under control of individuals the reason of a failings and external factors the reason for success.
Stereotyping	Generalize the type of individuals and ignoring differences.

Types of Rating Errors In this section we briefly discuss errors that can occur in ratings of performance [92]. These are listed in Table 3.8.

The best way to avoid rating errors is to provide appropriate training to the appraisers.

3.4 Summary

In this chapter we briefly introduced some preliminary concepts of personality and performance appraisal. We found that different personality assessment tests designed on different theories are available. The most prominent choices for personality assessment tests are MBTI, IPIP, NEO PI tests. Based on the review of the personality assessment instruments presented in this chapter, we selected suitable instruments to be used for this research.

The background knowledge on performance appraisal presented in this chapter also helped to develop performance appraisal form for software testers.

Chapter 4

Research Method

4.1 Introduction

From the argument presented in Chapter 1 and the review of relevant literature described in Chapter 2, we identified some research gaps in software testing. To address these gaps, a set of formal research questions is presented. We designed a series of research studies to address these research questions. We then ran these studies with a range of different participants and analysed the results we obtained. The research questions, research studies along with the details of participants and instruments used to carry out the research studies are described in this chapter.

4.2 Research Questions

The key question of this research is to find out whether or not personality traits have any influence on the performance of software testers. We did not find any direct answers to this from the existing research. However, the existing relevant literature showed some indication that personality traits might influence the effectiveness of software testers. This research was designed to find the influence of personality on the effectiveness of software testing. In designing our research to find the answer to the key questions, we identified a number of other gaps in existing software testing-related research.

We know the role of the software tester is different to other practitioners in the software industry and that it comprises of specific tasks in the software development process. However, the boundary of the tasks of this role is not obvious from the review of the literature. Even the definition of the role of "software tester" is not precise or wholly agreed upon by the community. There is no agreed upon description of specific software testing tasks. This limitation encouraged us to investigate and find out what software testers actually spend their time doing.

Our goal in doing this was to find the relationship between the personality traits of software testers with their tasks and performance. In order to find any such associations we needed to distinguish between different performance levels. However we found no standard way of assessing performance of software testers. From the review of the literature it is not clear how the performance of software testers is actually appraised in practice. This gap in the available body of research encouraged us to find out how the performance of software testers is appraised in practice. In order to use a standard method of performance appraisal in this research we also proposed a novel performance appraisal form for software testers, to be used by their managers.

In reconciling the issues described above we planned our research in small research studies to find the answers to the following key research questions that we identified:

- RQ1: What factors influence the effectiveness of software testers?
- RQ2: What do software testers do?
- RQ3: What personality traits are over-represented among testers?
  - RQ3.1: Are these traits different from other practitioners?
- RQ4: Do personality traits influence software testing performance?
  - RQ4.1: If yes, which trait(s) has the maximum influence?
- RQ5: How is the performance of software tester appraised?
- RQ6: How the performance of software testers should be appraised?

4.3 Research Process

We designed individual research studies to find answers to each of our research questions above. In total we conducted five research studies. The studies are briefly described below:

Study 1: What factors influence the effectiveness of software testers?

In this study we collected the opinions of software testing professionals about different factors that they think might influence the effectiveness of a software tester. The study obtained the views of testers on those factors from both individual and group perspectives. The study is aimed to address research question "RQ1: What factors influence the effectiveness of software testers?". The study is reported in Chapter 5 and Chapter 6.

Study 2: What responsibilities software testing include?

This study collected the worklogs of professional software testers to find practiced responsibilities (lists of unit tasks) carried out when doing software testing. The study also collected software testing-related job advertisements and open source bug descriptions to find descriptions of testing related tasks. The study aimed to answer the following research question "RQ2: What do software testers do?" and is reported in Chapter 7.

Study 3: Personality traits of IT professionals

This study collected the personality profile of professional software testers and other IT professionals to find whether any trait is over represented among either type of professionals. The study was designed to address research questions "RQ3: What personality traits are over-represented among testers?" and "RQ3.1: Are these traits different from other practitioners?" and is reported in Chapter 8.

Study 4: The effect of personality traits on effectiveness of software testing

This study collected the personality profiles of students and measured their effectiveness in performing some representative software testing tasks, to find the correlation between these if there is any. The study answered research questions "RQ4: Do personality traits influence software testing performance?" and "RQ4.1: If yes, which trait(s) has the maximum influence?". The study is reported in Chapter 9.

Study 5: Performance appraisal of software testers

This study collected the information about the state of practice of performance appraisal of software testers in industry. This study also collected the feedback of testing managers on a proposed performance appraisal form for software testers that we developed. This final research study addressed the final two research question "RQ5: How is the performance of software tester appraised?" and "RQ6: How the performance of software testers should be appraised?". The study is reported in Chapter 10.

A common process was followed in conducting all of the above research studies. The process is described in Figure 4.1.

4.4 Research Design

Research is designed based on the research question, hypothesis and the sample [99]. Denscombe [100] suggests that we consider three key questions to decide

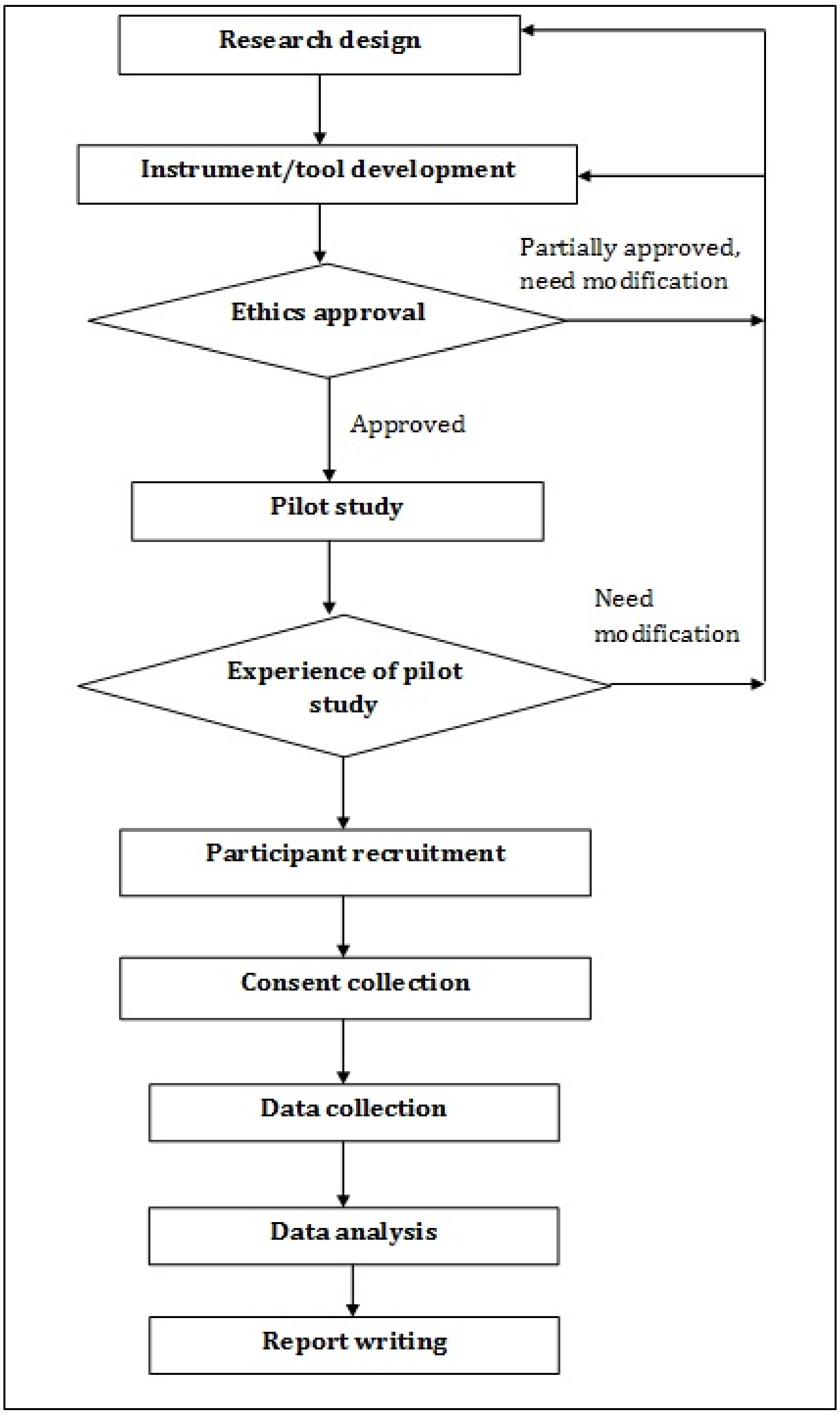


Figure 4.1: Research process

what strategy best fits the purpose of our research study:

- Is it suitable?
- Is it feasible?
- Is it ethical?

We evaluated different research strategies. A checklist for three key questions was prepared for different research strategies and the most appropriate strategy based on the specific research question(s) and the type of sample was selected. The strategies applied for our research studies are described below:

4.4.1 Survey

A survey is a research strategy that enables researchers to collect a large set of data in a timely manner. A survey usually has a wide coverage. According to Denscombe “to survey” is associated with the meaning “to look” [100]. There are different types of survey. We selected Internet surveys for our research studies. Internet surveys can be conducted through email or though web based questionnaire. We used web based questionnaires for our research studies when applying the survey research technique. Web based surveys are less expensive and less time consuming compared to the conventional paper based surveys. Since the intended participants of our surveys were software development related practitioners, we could assume they had easy access to the internet. As such web based survey was our first choice. At the cost of survey tool development, the other procedural delays could be mitigated.

4.4.2 Quasi Experiment

According to Denscombe [100]- “An experiment is an empirical investigation under controlled conditions designed to examine the properties of, and relationship between, specific factors”. The main advantage of conducting experiments is it can be replicated. Experiments can be true experiment, quasi experiment or correlational experiments [100]. We conducted a quasi experiment for one of our research studies. Compared to other experiments, the treatment is not randomly assigned to experimental subjects in a quasi experiment.

4.4.3 Case Study

According to Denscombe [100] the defining characteristics of a case study is its “focus on just one instance of the thing being investigated”. However, occasionally two or more instances can be considered. Unlike surveys, case studies do not have a wide spectrum. The focus is limited to individual cases.

4.5 Participants

There were two major groups of participants involved in our research studies: ICT students and software testing related professionals. For experimental research in software engineering, Juristo and Moreno [101] referred to the participants as “experimental subjects”. They suggested that researchers carefully address the role of the experimental subjects, since unlike most other disciplines, the results can differ depending on the influence of the experimental subjects in software engineering. We believe similar cases can occur in the other studies proposed by us. The type of responses we are seeking were highly dependent on the participants. For studies 3 and 4 we are interested in personality traits of the participants that takes individual variability into account. However, there are many other factors on which individuals can vary, e.g. experience in software testing. We expected to capture these variances through a demographic questionnaire. However capturing all of these variables is impossible given the length and scope of the studies we conducted. All of our research studies included demographic questionnaires and the information collected though these questionnaires was carefully considered when analysing data.

4.6 Recruitment

We applied multiple sampling techniques for recruiting participants for our research studies. Sampling is referred to selecting a subset of participants from the total available population. For many obvious reasons recruiting the total population to participate in research studies is not possible. Hence, we need to select a subset of participants called the sample from the population. The different sampling techniques used for the research studies reported in this thesis are described below.

Cluster sampling

In cluster sampling, instead of selecting individuals from the population randomly, clusters of individuals are selected and within one cluster all individuals are included in the sample [100]. In studies 1, 3 and 5 we used cluster sampling. In these studies we invited participants from different mailing lists. We selected the cluster of participants who are associated with mailing lists and groups. In comparison to sending invitation to individuals, sending email to the groups (clusters from the population) extended our ability to invite larger group of participants in shorter time.

Snowball sampling

In inviting participants from personal contacts for studies 2 and 3, we applied snowball sampling [100]. Snowball sampling is the process whereby a sample is selected through references. The student researcher, the principal and the associate supervisors invited participants from their personal contacts and requested the invited participants to nominate more participants.

Representative Sampling

For study 1, 3 and 5 we targeted a representative sample. A representative sample is a subset drawn from the population that includes all relevant factors and matches the proportions of the populations [100]. Representative samples are usually associated with quantitative data.

Exploratory Sampling

In contrast, study 2 and 5 involved preparing a list of software testing unit job responsibilities and validating a proposed performance appraisal form for software testers, respectively. Our samples for these studies were exploratory. According to [100], an exploratory sample is suitable for generating new ideas and/or gathering new insights or information. Exploratory samples are usually associated with qualitative data.

4.7 Instruments and Tools

We used six types of instruments and tools to conduct our different research studies. These are described below:

- Information statements
  - General information statements
  - Consent information statements for participants
  - Consent information statements for authorizing participants
- Personality assessment instruments
  - NEO PI-3 Test materials
    - NEO PI-3 test items
    - Answer sheets (S form)
    - NEO PI-3 norms
    - Your NEO summary
  - IPIP-NEO test materials
    - Test items
    - Feedback form

- Software testing performance assessment instruments
  - Software testing task specification
  - Performance Appraisal Form (PAF)
- Data collection tools
  - Survey and case study tools
    - Preliminary survey tool
    - Worklog collection tool
    - PAF validation tool
    - Personality assessment survey tool
  - Demographic questionnaire
- Advertisement tools
  - Leaflets
  - Posters
  - Emails
  - Tweets
- Information extraction scripts
  - Scripts for Yahoo! groups
  - Scripts for bug descriptions
  - Scripts for job advertisements

4.7.1 Information Statements

An information statement contains a brief description of the research study being conducted including the purpose, procedure and expected outcome of the research. We used two information statement documents: general information statement and consent information statement. Both of these information statements contained the purpose, procedure and details of handling data regarding our respective studies. The ethical considerations, details of ethics approval and confidentiality issues of the data were clearly stated in both types of letters. Participants could retain a copy of the general information statement, if they desired to.

The participants who agreed to take part in our research studies were required to sign a consent information statement. Consent to participate in the research studies were collected from the participants in three different ways: paper based consent, electronic consent and email consent. Paper based consent forms were collected for the study described in Chapter 9. The signed paper copies of the forms were restrained by the researchers. For the studies described in Chapters 5, 6, 7, 8, 10 consent information with a "I give consent" checkbox was presented in the index page of the data collection website. Checking the checkbox was considered as giving the consent to participate in the study. For the study described in Chapter 7, we required the consent from the participant as well as from their supervisor/manager/employer to authorize the participation. In this study, if participants or authorizing personnel considered sending a signed paper copy back to us infeasible, they sent us emails giving consent. The emails were retained by the researchers as a proof of consent.

The general and consent information statement letters used for all our research studies are presented in Appendix A.

4.7.2 Personality Assessment Instruments

We used two personality assessment tests in this research. We used a paper-based NEO PI-3 test and a web-based IPIP NEO test.

The NEO PI 3 is a standard, commercially available test, used for personality profiling. It comes with a test booklet of 240 multiple-choice items. Each of the items is usually a one line statement describing a person. The participants need to indicate their level of agreement with the statement describing them. There were five point Likert scale responses attached to each item. We used a hand answerable S (self rating) form of the test where participants could mark their responses to the corresponding items in the test booklet. Once participants completed the S form, the researchers performed number of validity checks (such as response to the validity check questions A, B and C and counting same response for consecutive number of questions). If researchers considered the responses valid, then the top layer of the answer sheet was removed and the scores of the corresponding responses were calculated from the bottom layer. The scores were then marked on the NEO PI-3 Norm forms and corresponding T scores were calculated. Based on the T scores the type of the participants were determined and marked in the "Your NEO Summary" form. The summary form contained all possible type descriptions. The type of the participants, as determined from the results of the test, was indicated with a tick mark in the summary form. In line with standard practice the researchers scored the test together with the participant.

We also used a short form of IPIP NEO test via a web-based survey instrument. This test contains 50 items describing a person. Since the study was web based, the collection of responses to the items, calculation of score and preparation of feedback was done with automated PHP scripts. The feedback was shown in a webpage. Feedback was also prepared as a PDF document that participants could download if they chose. This shorter, automated personality test allowed us to collect and analyse responses from a larger numbers of participants.

4.7.3 Software Testing Performance Assessment Instruments

We used two software testing performance assessment instruments in this research. The first one was designed as a software testing task by seeding faults in a Java program. The test was used for the assessment of effectiveness of student participants in software testing. Results of participants were scored using several factors to obtain a rating of their testing performance for this task. The test is detailed in Chapter 9.

Another performance assessment instrument was developed and proposed by the researchers to be used for performance appraisal of software testers in an industrial setting. The instrument implements a form based performance assessment of software testers. The details of the instrument and a study to validate the proposed instrument are discussed in Chapter 10.

4.7.4 Data Collection Tools

As discussed in Section 4.2, we applied many social research methodologies in this research. For the studies applying survey and case study we used web based data collection tools. Most of the studies had a custom web interface with relevant questionnaires, developed by us. The tools are discussed separately in Chapter 11.

Since all of our research studies involved human participants we used questionnaires to collect demographic information from the participants. Different studies required different participants; as such same demographic questionnaires were not used in all of the studies. The survey questionnaire are presented in Appendix B.

4.7.5 Advertisement Tools

We used several advertisement methods to invite participants for different studies. For most of the studies we advertised through software testing related Yahoo! and LinkedIn groups. We sent advertisement emails to those groups with the permission of the moderators. For study 4 we handed out leaflets to the student participants. In study 3 we attended Iqnite 2012 conference [102] to invite participants. We showed posters and handed out leaflets in the conference. For this study email was also sent to the YOW! email list by the organizers of the YOW! conference on request of the researchers. The organizers were very kind and they sent a tweet message with survey invitation to the tweeter.

4.7.6 Information Extraction Scripts

We implemented scripts to list software testing related Yahoo! groups and to extract the description of those groups from respective webpages. The scripts extracted the information automatically and stored in a file. The descriptions were then read by the student researchers to find out suitable groups.

We also implemented scripts for extracting job responsibility related information from the job advertisements of for software testers. A separate set of scripts were also written to extract bug description from bug repositories. All scripts were written in PHP.

4.8 Analysis Procedure

We collected both qualitative and quantitative data. We applied statistical analysis on the quantitative data where possible. The procedure of analysing both types of data is detailed in the following subsections.

4.8.1 Qualitative Data

Words, images, recordings all are considered as qualitative data. In our case, all our qualitative data were texts. We applied content analysis and grounded theory to analyse the qualitative data [100]. These are briefly discussed here.

Content analysis

The steps we followed while applying content analysis are:

1. Identifying the units of data: Each response to the particular question being analysed was considered a unit.
2. Form categories: The units of data were read multiple times and possible categories of responses were formed.
3. Coding units with categories: The units of data were then associated with a category.
4. Counting frequency: The number of unit responses within each category was counted.
5. Analyse the text: The categories are analysed to find relationships.

Grounded theory

The steps of grounded theory analysis of qualitative data are:

1. Exploring data: The researchers read the data multiple times to become familiar with the data.
2. Code the data: Codes were assigned to the data according to the interpretation made by the researchers.
3. Categorizing codes: Similar codes were grouped together to form categories.
4. Analysing codes and categories: The codes and categories were analysed to reduce their number and to develop a hierarchy.
5. Find key concept: Based on the analysis the key concept was found.

4.8.2 Quantitative Data

We applied different statistical analysis techniques on different quantitative data obtained from our studies, depending on the type of data and analysis we wanted to use. A number of our survey questions included Likert scale responses. The Likert scale responses usually represent ordinal data [103]. The responses were converted to numerical values (5 implies "strongly agree" and 1 implies "strongly disagree") for analysis. Table 4.1 presents the list of statistical procedures applied on the quantitative data.

4.8.3 Human Research Ethics

All of our research studies included human participants. The studies were anonymised, and as such no identifying information of the participants or the organizations where the participants worked or studied was stored. However, while we did not store explicitly identifying details, there was the possibility that enough information would enable to identify the participants. Therefore, we additionally checked for potentially identifying information.

The research goal and procedure were clearly stated in both the paper based and the web based consent information statement forms. All of the research studies requested enthusiastic and voluntary (except the study described in Chapter 9) participation from the participants. Participants were assigned unique codes after they signed (either on paper or electronically) consent information statement form. The participants were allowed to withdraw participation at any time, if they desired to. In the studies involving assessment of personality, participants were informed in the unlikely event that if it is required, counselling will be arranged for the participant. However, no participant sought counseling afterward.

Before conducting any research designed to collect data from people or to collect personal data about living people, it is important that the proposal be scrutinized by a committee of people of relevant expertise. In order to fulfil human research ethics requirements of Swinburne University of Technology we obtained approval from the Swinburne University of Technology Human Research Ethics Committee (SUHREC) before conducting every research study reported in this thesis. For the research study described in Chapter 9 we also obtained approval from Monash University Human Research Ethics Committee. The approval letters are given in Appendix C.

4.8.4 Pilot Studies

We conducted a pilot study with small number of selected participants before all our studies. The aim of the pilot studies was to identify and overcome the limitations of the proposed study designs before conducting those in field. All the web based survey links were given to the pilot study participants (mostly PhD students and some friends of the student researcher working in industry). The pilot study participants browsed the web sites and informed any problems they encountered or suggested modifications where they thought were important. The changes made based on the experience of pilot studies were minor. In the quasi experiment, the software testing task only was given to the pilot study participants. Based on their comment minor changes were made on the software testing task specification document.

4.8.5 Summary

This chapter describes the different research strategies adopted in our research studies, introduces our participants and lists the different instruments used in our research studies. Specific research studies are described in the following chapters.

Table 4.1: Quantitative data analysis

Statistics	Purpose
Frequency	Responses to the demographic questionnaire such as gender, nationality - number of responses to each category was counted.
Average	To find the middle value.
Standard deviation	To find the quantification of the broadness of the responses.
Kolmogorov-Smirnov Test [104]	To test deviation of population distribution from a normal distribution
Shapiro-Wilk Test [104]	To test deviation of population distribution from a normal distribution
Correlation	To find association between variables such as personality traits and effectiveness in software testing
MANOVA [104]	The multivariate analysis of variance was used for hypothesis testing. MANOVA was selected since there was more than one dependent variables.
Kruskal-Wallis [104] statistic	To test whether there is significant difference in median of responses to difference factors.
Tukey's Honestly Significant Difference (Tukey's HSD) [104]	Was used as post-hoc test. To test the significant difference of mean of responses to each pair of factors. This test was chosen when number of factors was large.

Chapter 5

Factors influencing effectiveness of individual software testers

5.1 Introduction

This chapter describes our first study that collected the views of professional software testers on the influence of the individual on the effectiveness of software testing. The study aimed to collect the opinions of testers on the importance of a variety of factors that influence effective testing, including experience, automation of testing, testing-specific training and different human characteristics. The study also asked about different factors that might be important for measuring the effectiveness of the tester. The study was conducted to address our first research question "RQ1: What factors influence the effectiveness of software testers?".

5.2 Methodology

We used a personal opinion survey [103] for our research. While there are a number of different techniques for soliciting information in social research, a survey was considered appropriate for this initial study as it allows the views of a wide range of people to be collected in a timely and practical manner [100]. Our survey was designed according to the six steps suggested by Kitchenham and Pfleeger [103].

5.2.1 Setting the Objectives

The survey was divided into two main themes - performance of individual testers and testing team development. This chapter reports the results relating to the first theme - performance of individual testers. The next chapter reports the finding on testing team development.

5.2.2 Survey Design

We used an online survey containing a self-administered questionnaire. The main advantage of using self-administered questionnaire is that the respondents can answer at their convenience.

5.2.3 Development of Survey Instrument

Kitchenham and Pfleeger [103] suggest searching for relevant literature before developing a survey instrument for two reasons. The first reason is to avoid duplicate research. The second is to learn and, if possible, to adopt questions and experimental design from existing relevant research.

We did not find any research into performance related factors of software testers in the existing research literature. However, relevant research investigating the influence of debugging skill and personality types [31; 32] , experience [11] and organizational issues [15] in software testing were helpful.

Our survey questionnaire included both closed questions (where the respondent chooses from a finite set of possible responses) and open questions (where the respondent provides a free-form textual response) [100]. Most closed questions used a Likert scale with five possible responses ("Completely disagree", "Somewhat Disagree", "Neither disagree nor agree", "Somewhat agree" and "Completely agree").

5.2.3.1 Questionnaire Design

The survey had a total of 29 questions, split into eight sections:

Personal Information: This section collected general information about the respondents including gender, age ranges, country and educational attainment. The survey was anonymous and as such did not record any personally identifying details of respondents.

Employment Information: This section contained general questions about the respondent's current role, responsibilities, and experience. We sought to determine whether they were testers themselves or interacted with testers, either as managers or colleagues. We also sought to determine the size of their employer in terms of number of staff, and whether the employer's primary business activity was software development or not.

Performance of Software Testers: Questions in this section asked the respondents about their broad view on what factors should be taken in to consideration to measure the performance of software testers, what human qualities influence performance in this role and what should be done, both individually and at an organizational level, to improve the performance of software testers. This section also asked whether programming skill and academic track record are connected with software testing effectiveness.

Automation of Testing: This section asked the respondents about their use and perceptions of automated testing tools. A review of the literature indicates that, to date, the majority of software testing research has been devoted to the enhancement of testing processes, test criteria, and to the development of new techniques and tools for different types of testing [6]. Underlying such research is the assumption that software testing should be, for the most part, a systematic, standardised and automated process. If so, then the key abilities required for a software tester are to be able to use the techniques, as implemented in automated tools - beyond that, the only insight and creativity required is to determine which technique or tool is most appropriate for the testing task. This view encouraged us to know what proportion of our participants use automated tools and in what ways tools help them.

Experience in Software Testing: This section asked respondents about their view of experience in software testing.

Characteristics of Software Testers: This section asked respondents' opinions on the personality characteristics of good testers. The characteristics listed were based on the well-known “Five Factor Model” of personality, which is one of the most popular models of personality traits in modern personality psychology research [73]. This model groups the many traits which can be used to describe an individual’s personality into five broad dimensions, including Extroversion, Agreeableness, Conscientiousness, Neuroticism, and Openness to Experience. The dimensions of this model were described, in simplified language, as possible characteristics of “good” software testers. The respondents were asked to consider a “good” software tester they personally knew and to indicate whether they believed that the individual considered exhibited these characteristics. An open-ended question giving respondents the option to mention other characteristics was also provided.

Training/Certification: This section asked respondents to comment on their experiences and opinions of training and/or certification. An active debate on the value of certification in software engineering [105; 106] encouraged us to know the perception of our participants about the importance of testing specific training and certification.

Test Team Building: So far, all the sections were asked questions from the individual tester’s perspective. The relative importance of many of these factors may vary while recruiting members for a test team. Different diversities may play an important role for testing team success. This final section asked questions about testing as a team activity. The results of this section are described in a separate chapter.

5.2.4 Evaluation of Survey Instrument

We administered a pilot survey on a small sample of software engineers. In this pilot survey the questionnaire was sent to seven software engineers requesting them to fill out the survey and to comment on the questionnaire. Five of them gave us their opinion about the questionnaire. Using their feedback, some minor changes were carried out, such as adding some additional factors and questions.

5.2.5 Obtaining Valid data

To get a sample population for the survey we searched LinkedIn and Yahoo! groups with the keyword "software testing" and listed the first 1000 groups. We read the description of the listed groups and selected 29 LinkedIn and 21 Yahoo! groups using purposive sampling [100]. In this sampling process we examined the description of each of the groups and selected those that were solely software testing related groups. We excluded inactive and specifically job vacancy-related groups. We sent a request to the moderators of these selected groups to allow us to invite group members to take the survey. Moderators of 12 LinkedIn and 12 Yahoo! groups approved our request. The group response rate was 41.4% for the selected LinkedIn and 57.1% for the selected Yahoo! groups. The numbers of member for Yahoo! and LinkedIn groups were 49786 and 39027, respectively. The individual response rate is unknown and can not be measured since a participant can be member of more than one group. Nor is it known how many group members actually read the group emails.

No financial or other reward was offered for participation. Respondents were offered the chance to provide an email address to receive a report summarizing the survey results. No association between the provided email addresses and survey responses was recorded.

5.2.6 Data Analysis

The Likert scale responses usually represent ordinal data [103]. The responses were converted to numerical values for analysis. Kruskal-Wallis and Tukey's Honestly Significant Difference (Tukey's HSD) tests were used for statistical analysis.

5.3 Results

A total of 104 respondents completed the survey. As the respondents had the option to leave any question blank, our results report how many respondents did not answer each question.

5.3.1 Demographic information

Majority of the respondents (around 70%) were male. Figure 5.1 shows that more than half of the respondents were between 18-40 years of age. Table 5.1 shows around 30% of respondents reported their "Country" as India, with the second largest group of respondents (around 25%) coming from the United States. The balance of respondents were from many different countries around the world. It was, however, notable that there were no respondents from several countries known to have substantial software industries, including (for example) China,

Table 5.1: Country (Question 2.2)

India	approx. 30%
United States of America	approx. 25%
Bangladesh	approx. 10%
Netherlands	approx. 5%
United Kingdom, Pakistan, Finland	approx. 3% each
Sweden, Romania, New Zealand, Brazil, Israel	approx. 2% each
UAE, Egypt, Switzerland, Philippines, Mexico, Poland, Austria, Australia, Denmark, Ireland	approx. 1% each
No Response	approx. 4%

Japan, and Germany. It is possible that there is a sufficiently large "critical mass" of testers in those countries to support testing mailing lists in their native languages. In the case of Brazil, the moderator of a Brazilian testing mailing list translated our invitation (but not the survey itself) into Portuguese, possibly explaining the two responses from that country.

Figure 5.2 indicates the educational attainment of the respondents. The vast majority had a university degree in software engineering (denoted SE in Figure 5.2) or another IT-related field, with a substantial minority possessing a graduate degree.

5.3.2 Employment Information

Figure 5.3 reports the employment status of the respondents. Nearly 60% of respondents were employed by large IT companies; with a little under 20% of the

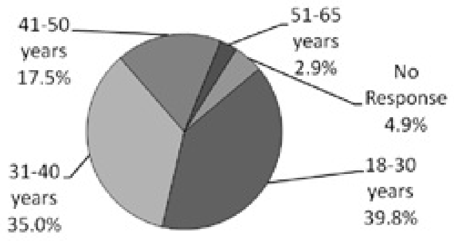


Figure 5.1: Age Ranges

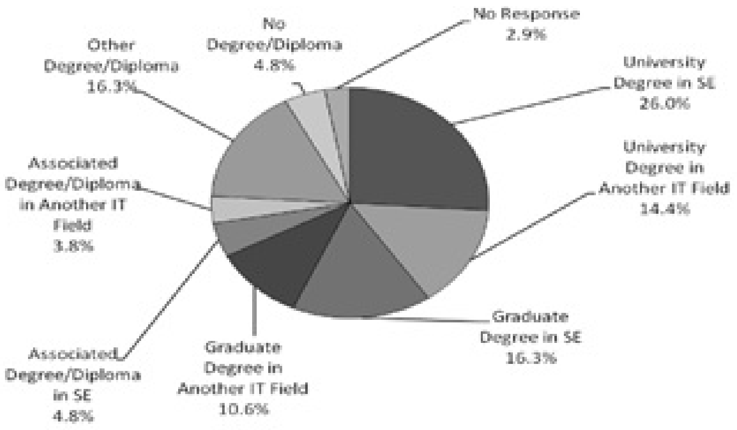


Figure 5.2: Educational Information (Question 1.4)

sample employed by smaller IT companies, and a similar proportion employed by larger "non-IT companies". Figure 5.4 shows that almost half of the respondents had more than five years of job experience.

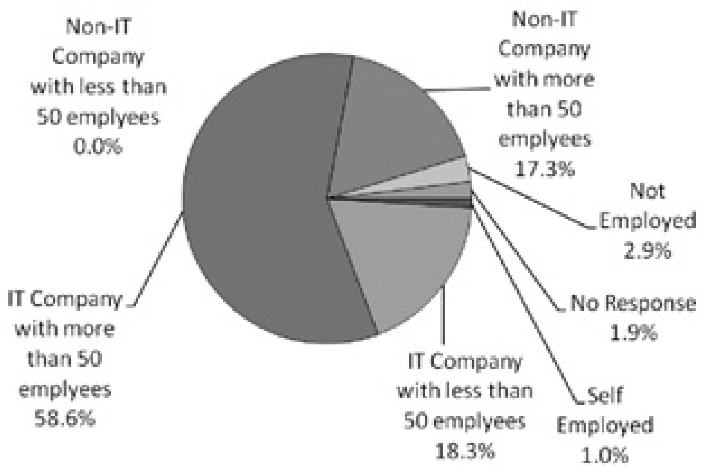


Figure 5.3: : Employment Type (Question 2.1)

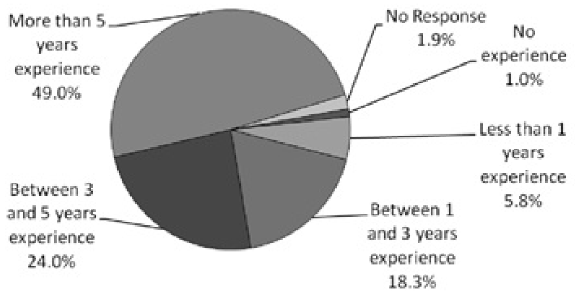


Figure 5.4: : Experience (Question 2.3)

Table 5.2 indicates the respondents' main job responsibilities. Note that respondents were able to select more than one option. More than 75% of respondents indicated that they were responsible for "testing software modules/pro-

Table 5.2: Main Job Responsibilities (Question 2.2)

Developing software module/program based on software specification and testing self developed modules/programs	approx. 8%
Developing software module/program based on software specification and testing modules/programs developed by others	approx. 12%
Testing software modules/programs developed by others	approx. 70%
Manage Software Testers within a project	approx. 45%
Others	approx. 20%
No Response	approx. 3%

grams developed by others" and around 45% of the total respondents indicated responsibility for managing "software testers within a project". Very few respondents indicated that they were responsible, either partly or primarily, for software development. Some respondents explicitly mentioned quality assurance (QA) management as their job responsibility. Individual respondents mentioned roles such as teaching, research, consulting, hiring testers, and business development, amongst others.

5.3.3 Performance

The vast majority of respondents agreed that performance of software testing varies from tester to tester, as indicated in Figure 5.5. Respondents believed that the difference was substantial, as reported in Table 5.3- approximately 35% of respondents stated that the best tester they had worked with was "50% more valuable to the project", and approximately 27% choosing an 80% figure. Interestingly, in open-ended comments, a number of respondents nominated higher figures, though a few were skeptical about being able to quantify the difference.

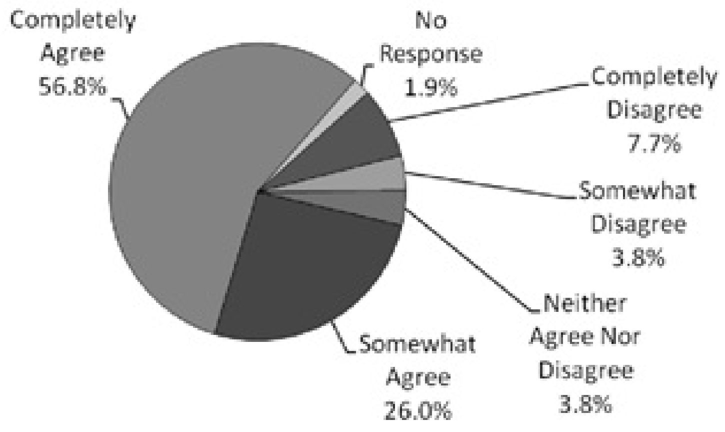


Figure 5.5: Responses to "Performance varies a lot from tester to tester" (Question 3.1)

5.3.3.1 Assessment of Performance

We listed five factors ("Number of bugs found", "Severity of bugs", "Quality of bug report", "Ability of bug advocacy" and "Rigorousness of test planning and execution") that we believed might be important for measuring the performance of the testers. Respondents indicated their level of agreement that these were important. The responses are shown in Figure 5.6. We can see that, except for "Number of bugs", the distribution of responses to the different factors were quite similar. The level of agreement is highest for "Bug report quality" and "Rigorousness of testing". A Kruskal-Wallis test showed that the perceived importance of the factors differed significantly ($p < 0.05$).

Table 5.3: Compared to an "average" tester, the best software tester you have worked with is... (Question 3.6)

20% more valuable to the project	approx. 10%
50% more valuable to the project	approx. 35%
80% more valuable to the project	approx. 25%
100% more valuable to the project	approx. 15%
Nominated other amount	approx. 12%
No Response	approx. 3%

Tukey's HSD tests were used to compare each pair of factors, showing that "Number of bugs found" was considered significantly ($p < 0.05$) less important than any of the other factors. Post-hoc testing also showed that "Quality of bug report" was considered significantly ($p < 0.05$) more important than "Severity of bugs found". However, there was no significant difference between other pairs of factors.

In the accompanying open question, around 30% of the respondents noted other factors as important, for instance, the quality of the communication with developers (approx. 10%), domain knowledge (approx. 7%) and understanding of requirements (approx. 5%). Other responses to the open-ended question included (in order of frequency of occurrence): analytical ability, implementation of plans, creativity, level of testing automation, and preventative teaching to the developers. Interestingly, one respondent suggested that the performance of a software tester can be measured by the number of bugs reported in the "live" environment (after deployment).

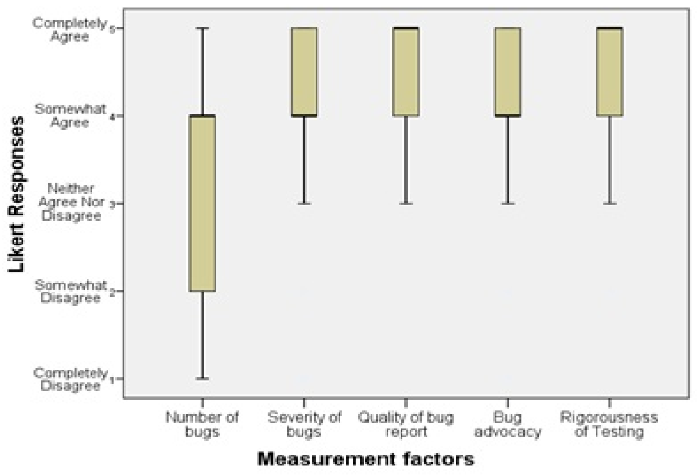


Figure 5.6: Responses on “Factors important in measuring performance of software testers” (Question 3.2)

5.3.3.2 Influence of programming skill and academic records

A majority of respondents did not think that programming skill helps to improve performance as a tester, with around 67% disagreeing, and less than 12% agreeing, as shown in Figure 5.7. As Figure 5.8 shows, there was a mixed response on whether academic records are a good predictor of testing effectiveness, with roughly half of the respondents agreeing to some extent, but 22% disagreed and 24% neither agreed nor disagreed. Since, no open-ended question was provided with these two factors, it is difficult to clarify the responses on these.

5.3.4 Factors that influence performance

We asked respondents to indicate whether they agreed that a number of different factors (“Knowledge of specific testing techniques”, “Expertise in the problem domain”, “Testing specific training/certification”, “Intelligence”, “Dedication”,

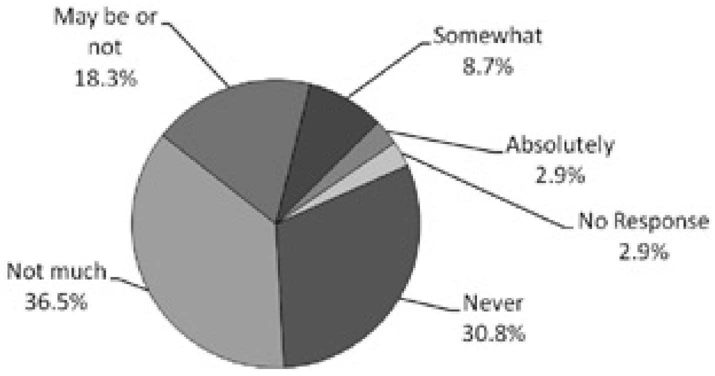


Figure 5.7: : Responses to question “Do you think good programming skills help to improve performance as a tester?” (Question 3.4)

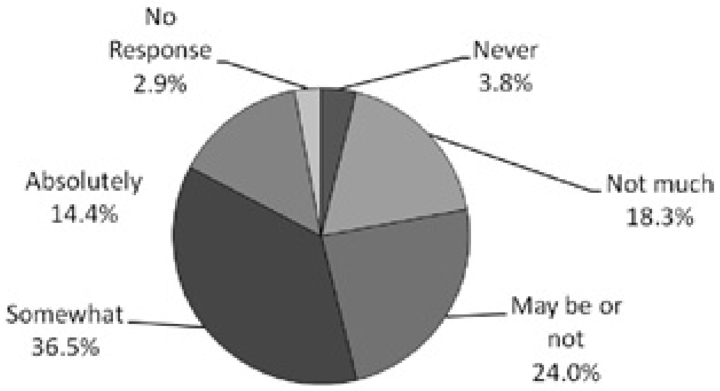


Figure 5.8: Responses to question “Academic record is a good predictor of performance of software testers” (Question 3.5)

"Punctuality/time value", "Thoroughness", "Positive attitude", and "Interpersonal skill") might influence the performance of testers. Figure 5.9 shows that most respondents agreed that these factors were influential.

A Kruskal-Wallis test showed a significant difference between the levels of agreement on the importance of the factors. Tukey's HSD test showed that the level of agreement that "Testing specific training/certification" and "Punctuality/Time value" were important was significantly less than for the other seven factors. However, there was no significant difference between these two factors, or among the other seven factors.

Some respondents (around 15%) mentioned other factors in the accompanying open-ended question. The most common one was "motivation" (approx. 2%); others included "Accepting new challenges", "Automation of testing", "ability to work under pressure", "knowledge sharing and good communication skill", and "out of the box thinking".

Individual measures for self-improvement Responses to the open-ended question 3.7 ("In your opinion, what can help to improve your performance as a tester") covered a broad range of ideas, from "*putting the evil hat on and trying to break the application in any way...*" to "*study philosophy, rhetoric, deconstruction, fallibilism, ethnomethodology, qualitative methods, grounded theory...* ".

However, some common themes were observed; for instance, testers must be dedicated and make an active effort to improve their work; as one respondent put it, testers should "love testing". Many respondents mentioned the need for learning, including both new testing techniques, and about the problem and business

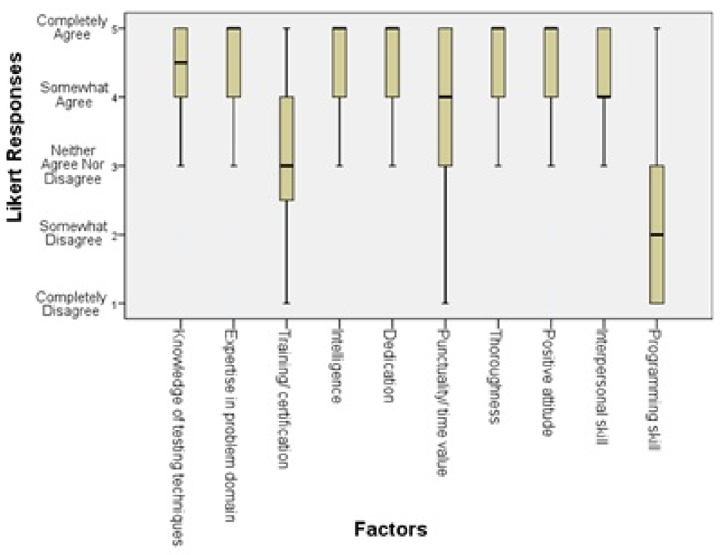


Figure 5.9: Responses on "Qualities Influencing Performance of Software Testers" (Question 3.3)

domain of their work. Learning from one's own personal experience, as well as the experiences of other testers, were also considered important by respondents.

Organisational measures for improvement Similarly, responses to the open-ended question 3.8 ("What can your employer do to improve your performance in your role of software testing") were quite broad. The most common theme amongst responses was the need for training, not only in testing techniques but also in the problem domain. Most of the respondents think that the employer should arrange training and ensure usage of new knowledge in the project. They also emphasised good communication with developers and customers (four of respondents specifically mentioned the importance of direct customer contact), including full access to product specification documents. Some respondents mentioned the importance of sufficient time for testing, and the need to introduce testing in the early stage of the development life cycle. Respondents believe that the employer should trust, respect, motivate, and encourage testers to do well in their job, and provide adequate recognition of good work.

5.3.5 Influence of Automated Tools

In response to Question 4.1, more than 60% of the respondents to our survey indicated that they frequently use automated tools for software testing, while 35% indicated that they do not. The most commonly-used tools reported by our respondents (Question 4.1.1a) were Selenium and QTP. The next most common type of tool mentioned was in-house custom testing tools. WinRunner, loadRunner, Jmeter, Fitness were also listed multiple times by the respondents.

Nearly one-third of respondents identified the most common benefit of automated tools was as a time-saver, though this sentiment was variously expressed as increased speed, improved productivity, and less manual testing effort. Related to this, around 15% of respondents mentioned that automated tools can help doing "repetitive and mundane tests" - presumably, this relates both to time savings and to reducing boredom. around 7% respondents mentioned that they could use the human time saved by test automation tools to perform additional testing. Some respondents mentioned the use of automated tools for specific types of testing - around 12% respondents mentioned that automated tools are especially helpful for "regression testing" - several of these specifically identified the ease of capturing test cases to run on updated software versions - while around 6% think these are helpful for load and performance testing. Around 6% respondents said that automated tools improved test accuracy. Other benefits identified by respondents included improved bug tracking and traceability. A small number of respondents stated that automated testing could improve the quality of testing, expressing greater confidence in tested code, and increased "reach", and that fewer bugs were found manually than with automated testing.

On the other hand, a small minority of respondents (approximately 5%) indicated reservations about automated tools. Most of the respondents of this group stated that automated tools often require excessive (and thus costly) maintenance. They also said that some automated tools sometimes produce large scripts. A respondent mentioned that most tools do not exactly do what one needs, so part of a tool can be used. Another respondent noted the lack of a "suspicious mind" in automated tools, and that they are a means, not an end in themselves: "*Machines can't feel, they can't have a hunch, they can't be suspicious, they can't investigate, and they can't change their minds due to better information. More important is that test automation shouldn't be a goal; test automation helps you achieve goals*". A few respondents mentioned that testers need to understand a tool very well before using it and should be able to judge when it is necessary or best to use automated tools. One also said that tools should be simple, quick and directly related to business value. A few responses suggested that testers should write their own tools after understanding the problem domain.

5.3.6 Experience

As Figure 5.10 indicates, a majority of respondents agree to some extent that performance grows with experience. However, this was neither universal nor unequivocal, with considerably more respondents choosing “somewhat agree” than “completely agree”, and a considerable fraction either disagreeing or declining to express an opinion.

We requested that respondents comment on the importance of experience in software testing (Question 5.3), and around 40% responded. While virtually all agreed that experience could be important, many expressed the view that not all experience is equally valuable. The following response is representative: “*Some people learn from experience, some don’t. Good testers become great over time; terrible testers stay terrible!*”.

Some respondents nominated specific reasons for the importance of experience. According to some of these, an experienced tester can easily get common bugs and can assess where the probability of bugs is high; as a result they can test new modules in quick time. One respondent also said “already tested test cases” remain in the mind of the experienced tester, presumably assisting in the planning and execution of future testing. According to the respondents, experience helps testers to prioritise work and is useful for better planning and analysis. The respondents also said experience helps to increase knowledge of the domain and the product. Some also said that experience helps to grow adaptability in different situations. However, one thing most of the respondents emphasised is that experience is only fruitful if testers learn from the past, including their mistakes. It was also mentioned that a variety of experience, including new challenges was important.

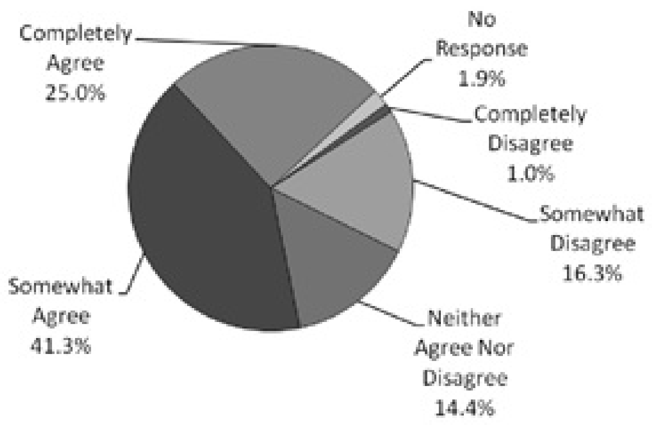


Figure 5.10: : Responses on "Performance grows with Experience" (Question 5.1)

5.3.6.1 Saturation of Experience

We asked whether the respondents think performance reaches "saturation" - that is, at some point, is there no benefit to additional experience. Figure 5.11 shows that respondents tended to disagree with this proposition, with only 21% agreeing and 44% disagreeing. It is notable that a high proportion of respondents either indicated no view, or did not answer the question at all.

For those who agreed that saturation of experience occurs, we asked when this point is reached, using an open-ended question (Question 5.2.1). The most frequent response to this was that it varied according to the individual. Some specifically said that saturation comes after 2-3 years - one respondent nominated a period of 5 years. According to some respondents saturation occurs if a tester is bored with repeatedly doing similar work. A few respondents stated that saturation can never come to the life of an IT professional until, as one respondent

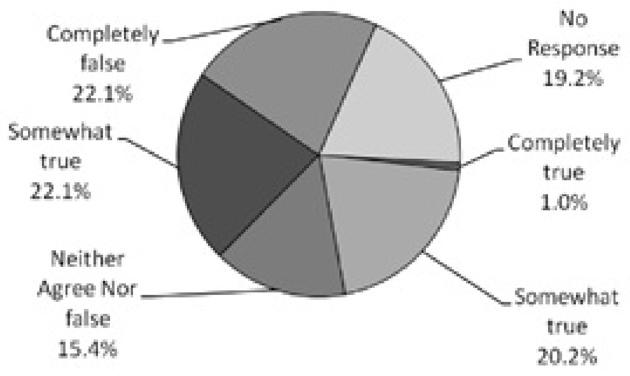


Figure 5.11: Responses on "Performance is Saturated after some experience"(Question 5.2)

put it, "after retirement".

5.3.7 Characteristics of Good Testers

This section listed human characteristics associated with the factors of the Five Factor Model of personality [74]. This is one of the most popular models in modern personality psychology research. The listed characteristics were "Good interaction with outward social world (Extroversion)", "Open mindedness", "Tendency towards negative emotionality", "Qualities like trust, modesty" and "Personal organization". We requested that the respondents consider a good software tester they have worked with (or themselves, if they believed that they are a good tester), and to state whether they agreed that the listed characteristics of this section were exhibited by that person. Figure 5.12 depicts the responses to this. The level of agreement was highest for "Open mindedness" and "Personal organization".

A Kruskal-Wallis test indicated that there was a significant difference ($p < 0.05$) in mean responses. Post-hoc Tukey's HSD tests show that "Open mindedness" was significantly ($p < 0.05$) more strongly identified with good testers than the other four characteristics, and "Negative emotionality" was significantly ($p < 0.05$) less strongly identified than the other four characteristics of good testers. No other differences were statistically significant.

In the accompanying open question, some other characteristics were also reported by some respondents (27%). Several of them said "attention to the details" (around 3%) and "innate investigation traits" (around 3%) should be characteristics of a good tester. Some other characteristics listed by the respondents include "scepticism", "tenacity", "loyalty", and "creativity".

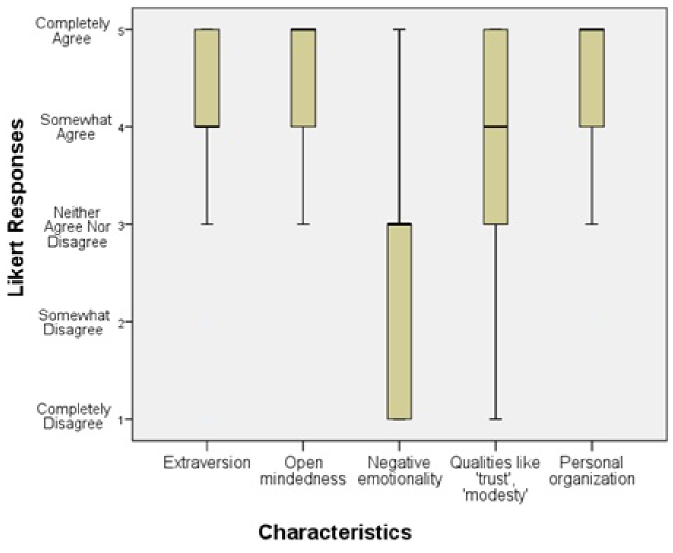


Figure 5.12: Responses on "Characteristics of good software Tester" (Question 6.1)

5.3.8 Training/Certification

More than 50% of the total respondents indicated that they had done training/certification in software testing in the last five years (Question 7.1). Of those, 78% mentioned the name of the training/certification. The most common named courses were the various levels of ISTQB [107] certification, representing around 30% of those who nominated a specific training course or certification. Some respondents said they have done training on scripting, QA tools and domain specific product knowledge. A few respondents mentioned attending conferences, and reading books and blogs.

We obtained a range of responses to Questions 7.2 and 7.3, which asked respondents to indicate whether they found the training/certification useful, and why. The majority found them at least somewhat useful. A number of respondents indicated that the certification courses were quite general and theoretical; a few mentioned alternative sources of ideas which they found more useful. Some respondents thought that their training/certification helped to better understand their work, provided new approaches to testing, and helped to save testing effort. These respondents believed that the training taught them techniques that broadened their "toolbox" - it was then their job to learn how to apply these in their specific problem domain. On the other hand, some respondents stated that the generic tools that are taught are often useless while self learned materials are considerably more useful.

5.4 Threats to Validity

We have considered two types of possible threats that can attenuate the validity of the survey outcome, which we discuss below:

5.4.1 Internal Validity

Misinterpretation of survey questions represents a threat to internal validity, and we believe this may have occurred with some questions. In the case of the characteristics section, we suspect that the underlying purpose of the questions was simply not understood by many respondents.

Secondly, as training and certification are perceived quite differently by some in the community of interest, and our questionnaire did not distinguish between the two, the responses on the questions of this section may be heavily influenced by the ongoing debate about the value of various certifications, rather than the value of ongoing specialized training more generally. Therefore, due to the likelihood of confusion in the responses to these two questions, any conclusions from these two sections of the survey could not be reliably drawn. It is, of course, possible that other survey questions were not interpreted as we had intended, but the results show no evidence that this was the case. For many questions the themes in the open-ended responses showed that the respondents had understood our intentions.

Another threat to the validity is the possibility of random (or just ill-considered) or less than candid responses, which is a common issue in this kind of study. However our survey responses were volunteered freely without any possibility of compensation, and were completely anonymous. Contact with potential respondents was made indirectly through broadly distributed mailing lists, rather than through individual contacts. Therefore, we believe that there would have been little motivation for either "throwaway" responses or lack of candour. The effort made in responding to open-ended questions by the respondents is also inconsistent with this threat.

5.4.2 External Validity

One possible external threat to the validity of the survey outcome is the representativeness of the respondents. As a voluntary survey with an unknown response rate, the survey does not represent any kind of random sample. Sampling through a limited number of mailing lists raises the possibility that the respondents may belong to certain subgroups within the wider testing community, with similar interests, experiences, and attitudes towards testing, which may not be reflective of the broader community of software testers. Responses were skewed towards testers from countries where English is the main language used in technical contexts.

Another threat to the external validity of our research is that it seeks only the thoughts and views of expert testers. They may reflect the "common wisdom" of the profession, but that common wisdom may well be wrong. It is routine in the physical and social sciences for specific empirical studies to reveal "common wisdom" to be unsupported by evidence. As such, while valuable, we cannot claim our results are conclusive. Instead, our survey may best be thought of as a source of plausible theories which can be justified with more detailed empirical studies.

5.5 Analysis

In the opinion of our respondents, the effectiveness of individual software testers varies considerably, as discussed in Section 5.3.3. While it would be unwise to over-interpret the quantitative estimate provided by our respondents in this section, their responses to this alone indicate that testers themselves strongly believe that the individual matters a great deal. We can therefore go on to consider what influences the individual's effectiveness, and whether these influences are innate, or the result of training, experience, or other external factors.

Our finding that "Number of bugs" was least important of the stated factors in measuring the performance of testers is consistent with Kaner [64], who suggests that the effort a tester puts to find harder bugs is not counted if we only look for bug counts. "Quality of bug report" is also considered important by Kaner. However, the outcome that "Quality of bug report" is significantly more important that "Severity of bugs" is, to our knowledge, novel, and if replicated sufficiently has significant implications for both the assessment and training of software testers.

Although there were significant differences among the responses on different factors that influence performance of testers, our data does not reveal what factor(s) are most influential. However, the tests suggested that "Testing specific training/certification" and "Punctuality/Time value" were considered less important. We were surprised that “Punctuality/Time value” was considered of lower importance, as punctuality is a desired quality in general and testing is mostly done to a tight schedule. Similarly, we were surprised by the finding that “programming skill” was considered less important, and have no explanation for this result. In both cases we believe that these counterintuitive results should be further investigated.

The difference in responses about the characteristics of testers indicates that there are characteristics that are common in good testers. Most notably “Open mindedness” was considered desirable for good testers. Since the characteristics in the question were based on the Five Factor Model of personality, our results suggest that certain personality traits are associated with good testers. ’Da Cuhna and Greathead [31] found that individuals identified as “Intuitive and Thinking (NT)” by the Myers-Briggs Type Indicator (MBTI) perform well at debugging tasks. “Openness to experience” of the Five Factor Model is positively correlated with the MBTI type “Intuitive”, so our results are consistent with Cunha and Greathead in this respect. We believe that our results provide additional justification for further investigation of the connection between specific personality types and performance in testing. Their research was based on MBTI, which is based on a very different model of personality, and due to sample size limitations they were only able to examine a subset of the MBTI factors. Due to these differences and limitations, it is difficult to directly compare our observations to their results. However, “Intuitive and Thinking” sounds quite similar to “open minded”.

As shown in Section 5.3.5, the majority of survey respondents did use some kind of test automation tool. These speed up testing, improve accuracy, and free testers to devise new tests, rather than conducting repetitive and prosaic tasks. As Kaner put it, the tools extended human reach [7]. Tools were considered particularly useful for regression testing. However, it is important to consider the nature of the tools used by our respondents. QTP and Selenium automate the process of test execution and evaluation, and assist with the test and defect management process. They do not automate the generation of test cases. Nor did testers mention the use of test coverage tools to evaluate test adequacy. The main use of automated test case generation was for performance testing (through the use of LoadRunner and Jmeter), not for functional testing. This may be an artefact of our sample, as it is possible that test generation tools are popular in particular problem domains, but not in those in which our respondents work.

Another possibility, in the case of coverage tools, is that unit testing, where coverage is most significant, may be performed by developers themselves rather than a distinct testing team. It is also possible that the in-house custom tools mentioned by a number of respondents were used for test case generation. Our findings on automated tools were similar to those of Ng et al. [108], who conducted a survey of Australian software testing practices in 2003. They found a similar proportion of survey respondents used automated tools; furthermore the tools were used primarily for automating test case execution, regression testing, and defect tracking. It is striking that, seven years hence, the parts of the testing process which are automated have not changed. If test generation tools are indeed not widely used in industry, this presents a challenge to academic testing research: why haven't automated test case generation tools found their way from the laboratory to industrial use?

Experience in software testing was considered important by most of our survey respondents. However, respondents noted that experience can be fruitful if testers learn from the experience. Most of the respondents also mentioned that experience can not be saturated.

In general, our respondents tend to believe that certification and training is helpful, but there were a variety of views as to the type of training that is most helpful. It is unfortunate that our survey did not distinguish between training and certification, which limits our ability to draw strong conclusions about this area as noted in Section 5.3.8. Certification has a dual role - as well as an opportunity for learning, certification provides a credential indicating that the holder has (presumably) demonstrated an understanding of the syllabus, which may be significant for recruitment purposes. There is a long lasting and vigorous debate on the value of certification in software engineering [105; 106], which we did not intend to contribute to with this survey. However, it is notable that some respondents found their training/certification to be quite abstract. According to them, the materials taught, most of the times, were not directly applicable at the workplace. Some found this a useful theoretical basis for their work, others reported difficulty in applying what was taught in practice. In response to the questions on self-improvement and employer assistance, training in the problem and business domain, as well as in specific technologies, were frequently mentioned.

5.6 Summary

In summary, our survey responses indicate that, according to professional testers, testing performance does strongly depend on the tester, and furthermore the perceived variance between testers is large.

For the measurement of existing testers, a number of factors were suggested by our respondents. One clear finding was that bug count is not considered a good measure of performance. These findings helped us to design a novel performance appraisal form for software testers. The responses also indicate that, what makes a good measure of testing effectiveness needs to be investigated further which we address in our fifth study reported in Chapter 10.

The responses indicate that there are factors that influence the performance of testers. However, our study could not determine precisely what the factors are. We also found that certain personality characteristics, particularly "open-mindedness", are perceived to be associated with good testers. This gives us good ground to design research studies investigating the influence of personality on the effectiveness in software testing.

In the following chapters we report more research studies to investigate these issues in more depth.

Chapter 6

Development of effective test teams

6.1 Introduction

In Chapter 5, we discussed about the effect of different factors on the effectiveness of testing. The importance of these factors was considered from individual perspective. However, while developing an effective test team, different factors might be considered important. Unfortunately, the question of how best to develop an effective testing team - which may or may not be the same as finding the best individual testers - has received surprisingly little research attention to date.

In addressing RQ1 we designed the research study reported in Chapter 5. However, we realised the relative importance of the factors can vary while selecting members for a test team. So we extended the survey and added the second theme.

In this chapter, we present the results of the second theme of our preliminary survey that has already been discussed in Chapter 5. This theme of the survey examined what factors might be important for developing testing teams.

6.2 Methodology

We used a personal opinion survey [103] for our research.

Our survey was designed according to the six steps suggested by Kitchenham and Pfleeger [103] and was described in more detail in Chapter 5. The steps of designing this theme of the survey are described in the following sub sections.

6.2.1 Setting the Objectives

The main objective of this second theme of the survey was to identify the most important factors that determine the performance of software testing teams. We specifically examined whether diversity was important for a testing team. If so, what diversity did participants consider the most important. We also investigated whether experience working as a team was considered important to performance.

6.2.2 Survey Design

We used an online survey containing a self-administered questionnaire, since these type of questions help the respondents to answer at their convenience.

6.2.3 Development of Survey Instrument

We did not find any research on development of software testing team in the literature. Existing research of the influence of different factors on team development (not specific to software testers) [109] led us to hypothesize that there were multiple factors that could influence testing team performance. Relevant research investigating the influence of different factors on the performance of IS teams was also helpful [110; 111**?**]. Factors like diversity of personality among team members [110**?**], experience [111], team playing capability and communication skill were included in the survey. These factors are considered to be influential on the performance of an IS team. We speculated they may apply to testing teams as well. The relative importance of these factors, moreover, would vary. From our previous experience, a review of the team-development literature, and the feedback of a pilot survey, we listed seven factors - “Testing performance (performance of individual members)”, “Interpersonal skill”, “Team playing capability”, “Experience in testing”, “Training/certification in testing”, “Knowledge of specific problem domain” and “Compatibility with other proposed team members (if known)” - that might be important for developing a successful testing team.

The importance of different types of diversity within a team has also been studied in general team development research [112]. We listed four types of diversity - “Diversity of personality”, “Diversity of professional background/experience”, “Diversity of age” and “Diversity of communication skill’ - based on our experience and review of the team-development diversity literature.

The survey questionnaire included both closed and open questions. Most closed questions used a Likert scale with five possible responses (“Completely disagree”, “Somewhat Disagree”, “Neither disagree nor agree”, “Somewhat agree” and “Completely agree”).

6.2.4 Questionnaire Design

The survey instrument contained six closed and two open questions split in to three main sections.

Important Factors for Developing Testing Team:

This section had three closed questions. The first question asked the participants to rank seven factors described in Section 6.2.3 according to their importance for recruiting team members. The participants could assign a rank from a range of 1 to 7 to each of the factors in ascending order (lower number implies higher rank). They could also assign the same rank to more than one factor, if they considered the factors equally important. An accompanying open question asked participants to list more factors that they think are also important. The second and third questions asked the participants whether they think all members of a testing team should be good testers and good team players. Both questions were closed and had Likert scale responses available with them. The second and third questions asked the participants whether they think all members of a testing team should be good testers and good team players, respectively.

Diversity of the Testing Team:

This section had two closed questions. The first question asked participants whether they think diversity in a testing team helps to improve performance with two possible responses- “Yes” and “No”. The second question asked participants what diversities they look for developing a testing team. The question had four types of diversity listed as described in Section 6.2.3 and each of the diversities had 5-point Likert scale choices attached to them. Participants could also report other types of diversity (not present in our list) that they considered important, via an open question.

Experience of the Testing Team:

This section contained one question that asked participants whether they think that a testing team performs better when they have experience working as a team. There were Likert scale responses for this question.

6.2.5 Evaluation of Survey Instrument

We administered a pilot survey on a selected sample software engineers (described in Chapter 5). Depending on the feedback of the participants of the pilot survey, minor changes such as adding some additional factors and questions were carried out.

6.2.6 Obtaining Valid Data

We used purposive sampling to select sample population for our survey. The details of sampling were discussed in detail in Chapter 5.

6.2.7 Data Analysis

The Likert scale responses usually represent ordinal data [103]. The responses were converted to numerical values (5 implies "strongly agree" and 1 implies "strongly disagree") for analysis. Kruskal-Wallis and Tukey's Honestly Significant Difference (Tukey's HSD) tests were used for statistical analysis.

6.3 Results

Total 104 participants completed the survey as mentioned in Chapter 5, however 4 of them did not provide response for the testing team developing questions. Therefore, 100 responses are reported here.

6.3.1 Demographics

The majority of the respondents (73%) were male. 76% of the respondents were between 18-40 years of age.

60% of respondents were employed by large IT companies; with 18% of the sample employed by smaller IT companies, and a similar proportion (17%) employed by larger non-IT companies. Large company was defined as having more than 50 employees. An IT company was defined whose main business product or service is related to IT. Among the participants 77% reported their job responsibility includes testing software/programs developed by others. 45% reported that they manage testers. Respondents were able to select more than one job responsibility. Half of the respondents (50%) had more than five years of job experience.

6.3.2 Important Factors for Developing testing team

Figure 6.1 depicts the ranking of the importance of the seven identified factors for effective testing teams. To improve the clarity of the graph we have reversed the ranks so that a larger number indicates higher rank (and thus importance).

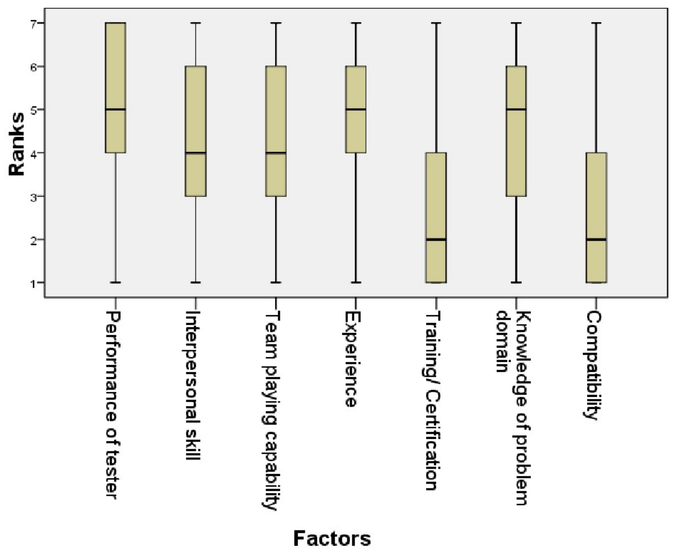


Figure 6.1: Rank of important factors (Question 8.1)

Some participants (25%) listed other factors that they consider important while developing a testing team via the accompanying open question. Among them, 3% of participants suggested that, they look for “Learning ability”, 2% suggested that they look for “Programming skill”, “Analytical skill” and “Communication skill”.

A Kruskal-Wallis test ($p < 0.01$) showed there is a significant difference between the mean rank value of the factors.

We have used Tukey's HSD as Post-hoc test to find which factors' mean ranks differ significantly. The test results indicate that "Training/certification in testing" and "Compatibility with other team members" are ranked significantly ($p < 0.05$) lower than all other factors. "Experience in testing" is significantly ($p < 0.05$) ranked higher than "Interpersonal skill".

The agreement with the statement "All team members should be good testers" and "All team members should be good team players" are showed in Figure 6.2 and 6.3, respectively.

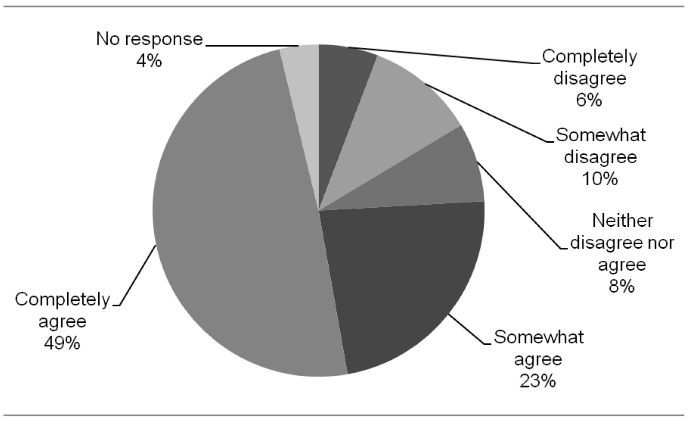


Figure 6.2: Level of agreement on whether all members should be good team players (Question 8.2)

6.3.3 Diversity of the testing team

89% of the participants agreed that diversity in a testing team helps to improve performance. 9% of participants suggested the opposite. 2% of participants did

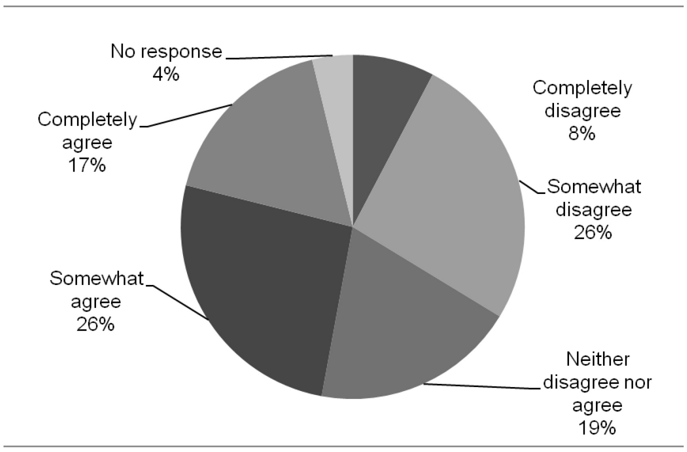


Figure 6.3: Level of agreement on whether all members should be good testers (Question 8.3)

not respond to this. Figure 6.4 shows the distribution of responses for the four types of diversity.

In the accompanying open question, 19% of participants listed other types of diversity that can be helpful for a testing team. The responses to the open question included (in order of frequency of occurrence): “Cultural diversity” (4%), “Knowledge of diverse domain” (4%), “Gender diversity”(3%) and “Diversity of academic discipline” (3%).

A Kruskal-Wallis test indicated that there is a significant (p < 0.01) difference in the influence of different types of diversity.

A Tukey’s HSD post-hoc test revealed that “Diversity of professional background/experience” was considered significantly (p < 0.05) more influential than the other three types of diversity. “Diversity of personality” is significantly (p < 0.05) more influential than “Diversity of age” and “Diversity of communication skill”. There was no significant difference between “Diversity of age” and

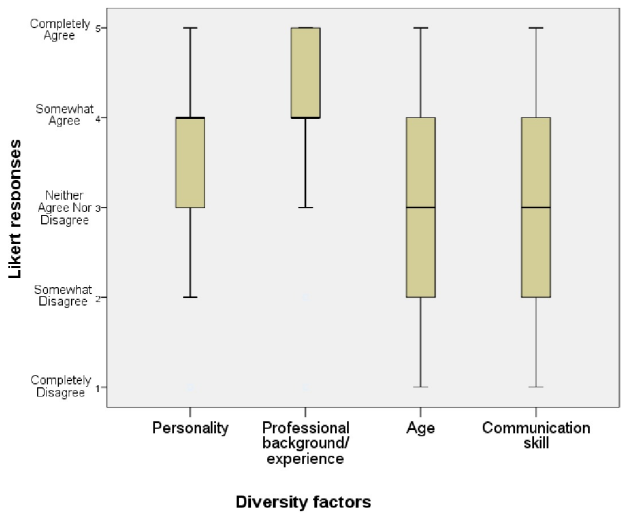


Figure 6.4: Level of agreement on different type of diversity (Question 8.5)

"Diversity of communication skill".

6.3.4 Experience of the testing team

Figure 6.5 shows the level of agreement of the participants to the statement "A team performs better when they have experience working as a TEAM". The responses indicate that 7% of participants at least somewhat agree that a team performs better when they have experience working together.

6.4 Threats to Validity

Misinterpretation of the survey questions by respondents threatens our study's internal validity. However, the results show no evidence for this. Another potential threat is random or less than candid survey responses, which is a common

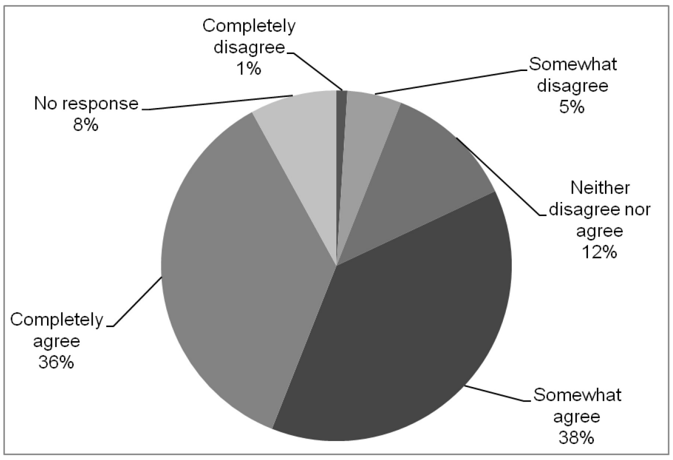


Figure 6.5: Level of agreement on "A test team performs better when they have experience working as a TEAM, rather than gathering experience as individuals" (Question 8.6)

issue in this kind of study. We see no evidence that it occurred in our data, and no particular motivation for participants to do so.

Since the individual participation response rate is unknown, we cannot claim much confidence on the representativeness of the respondents. Our research seeks only the thoughts and views of expert testers that may reflect the "common wisdom" of the profession, however that common wisdom may well be wrong. The possibility of a response bias, based on the construction of the questionnaire, is also a plausible threat to the external validity of our research.

6.5 Discussion

The results suggest that "Testing performance", "Experience in testing" and "Knowledge of problem domain" are the most important factors to be considered while developing a testing team. This is consistent with the observations of Beer and Ramler [11], that reported- experience resulted in higher domain knowledge that helped testing in case of insufficient or inaccurate specifications. However, the perceived benefit of experience was observed and reported for individual testers and not for a team. The importance of "Testing performance", either for individual tester or for team developing, to our knowledge, has not been supported by any research to date. The higher importance of this factor according to our survey suggests research is needed to investigate the influence of this factor in more detail.

Statistical test results reported in Section 6.3 indicate that there is a difference in importance among the seven factors for developing a testing team. However, we cannot prepare a simple ranking the seven factors from the responses since the differences are not significant for all pairs of factors. However, we can conclude that "Training/certification in testing" and "Compatibility with other team members" are considered significantly less important than all other factors. The responses to "Training/certification in testing" are not surprising, as the benefits of certification in software engineering are not supported by all [106]. However, responses to "Compatibility with other team members" are not consistent with Schutz's hypothesis that states compatibility has a positive influence on team productivity [113]. This raises the question of why compatibility is not given much importance compared to other factors, in a testing team. Further research is required to substantiate this result and shed light on the reasons behind it.

Our results also suggest that "Experience in testing" is significantly more important than "Interpersonal skill". Again the importance of experience is consistent with the observations of Beer and Ramler [11] (as has already been discussed above). However, the relative importance of these two factors according to our survey suggests research is needed to investigate the relation in more detail.

According to the responses to the closed question asking whether diversity of a testing team is helpful or not, we can conclude that diversity is considered helpful for a testing team to perform better since almost 90% participants selected the option “Yes”. Our next goal was to find what type of diversity participants think is helpful. The distribution of the responses to different types of diversity and our statistical tests of the responses suggest that not all diversities are equally important. Specifically, “Diversity of professional background/experience” and “Diversity of personality” seem to be more important than the others. The influence of “Diversity of personality” on IS team development has been supported by [56] and [13]. Here the authors of the studies suggest different IS roles for people of different personality types. However, our finding that “Diversity of professional background/experience” is considered influential is, to our knowledge, new and needs to be investigated further.

The majority of the participants tend to agree with the statement that “A testing team performs better when they have experience working as a TEAM, rather than gathering experience as individuals”. However, the confidence of the agreement seems to be low, since a greater number of participants “somewhat agree” with the statement than the number of participants who “completely agree”. One possible interpretation is that practitioners believe that team experience makes some contribution to performance, however other factors are of greater importance.

6.6 Summary

Our study indicates that respondents consider multiple factors important for developing testing team.

It indicates that some factors are perceived as more important than others, though the differences did not often reach statistical significance. We also conclude that diversity is desired for a testing team. “Diversity of professional background/experience” and “Diversity of personality” are two important types of diversity to be considered for developing testing team. However, the specific influence of these on testing team development and performance needs further empirical study. We also found that the practitioners consider experience as a testing team is helpful for better performance, although not as important as other factors.

Chapter 7

Job responsibilities of software testers

7.1 Introduction

As discussed in Chapter 2, an empirically validated list of unit tasks performed as part of software testing is not currently available. This lack of empirical data on software testers' unit job responsibilities impedes research on finding who is best suited for what role. For example, without information about what testers actually do, any research assessing who might be best suited, or for what testing specialisations, is just speculation. To fill this gap and to find an answer for "RQ2: What do software testers do?", we designed a research study to identify the unit tasks of software testing and to prepare a list of job responsibilities practiced by software testers. This chapter presents the details of the study.

7.2 Methodology

The research is designed to find answer to “What unit tasks software testing include?”. In order to answer the research question we planned to collect information about the unit tasks of software testing. One source of this information is the collection of work logs of software testers. In broad terms, a worklog is collection of units of work performed with a tracking of time spent to perform the unit of work. The analysis of worklogs of instructional design professionals [114] is an example of this type of research. However, such information collected with this method is dependent on the organization the software testers work in and the phase of the project they are involved in. Different organizations may define very different set of responsibilities for testers. Also, different levels of testing effort may be required at different phases of the project. Depending on the level of required testing effort the role of the tester can vary. Thus, information collected in this fashion may not reveal the whole picture.

Another source of this information is the descriptions of software testing roles in job advertisements. Recruiters generally prepare a list of responsibilities to be carried out by the newly recruited person and give the list in the job advertisement so that interested candidates get an idea of the role. The job advertisements for software testers typically contain lists of such responsibilities to be carried out by recruited software testers. The job advertisements for software testers can thus be a very good source of the soft of role information we are looking for. However, the responsibilities listed can be dependent on the business and work pattern of the organization. Collecting number of job advertisements from number of organizations can help overcome the problem. The research studies of Capretz and Ahmed [13; 14] encouraged us to consider this source of information. They collected the job responsibilities mentioned in different job advertisements for different software engineering roles, including software testing, and connected those responsibilities with some human skills that they believed would be helpful to carry out that responsibility.

Another possible source of information is bug repositories. Bug repositories are the collection of descriptions of the bugs encountered in a software development project. Bug repositories contain bug reports. These often contain enhancement requests as well. Bugs are often found while testing the software. Thus, the information about what the reporter did to produce the bug actually describes the tasks performed by a tester to find a bug.

Since such different sources of information were considered for this research study we chose triangulation [115], which refers to using more than two research methods. We analysed different social research methodologies [100] and adopted two different methodologies (Survey and Case study) to collect information from the different sources of information described above. We initially selected survey as our research methodology to collect information from different sources. However, due to very low response rate in worklog collection we analyzed the obtained data as is done in case study methodology. Therefore, we report the collection of worklog using case study methodology.

We also selected different social research strategies [100] to help collecting data. Table 7.1 describes the different methods and strategies followed for this research along with the rationale of selecting the method and strategy.

Table 7.1: Description of research methods and strategies

Method	Strategy	Rationale
Case study of software testers' work	Questionnaire	In this method we are expecting to collect detailed information on what testers are doing. The detailed nature of the data does not encourage many participants. As such a case study was performed with small number of participants. Case study is suitable for the type of information we expected to record. Since observing cases at their work for two weeks was not feasible we used web based questionnaire to collect the worklog.
Survey of software testing job advertisements	Observation	There are many job descriptions available. Due to the quantity of data survey was chosen. Survey allows collecting huge amount of data in small amount of time. Since this method did not involve any human participants, no questionnaire was used. The collected job advertisements were observed to get necessary information.
Survey of bug descriptions in bug repositories	Observation	Similar to the second approach, there are lot of bug description available from the bug repositories. A survey helped us to collect the bug description in less time. The bug descriptions were then observed to extract information related to testing responsibility.

7.2.1 Case Study of Software Testers' Work

In this method we collected the worklogs of different software testers. The details of this method are described in the following subsections.

Case selection:

The participants of this study were recruited from the personal contacts of the researchers. The researchers also contacted few software companies who have some software testers employed in it. The testers were requested to participate in the study through their employer company. This was a voluntary participation request and had no connection with their employment. The employer did not have any access to the responses.

Procedure:

Over a period of approximately two weeks, testers indicated (in a broad sense) the nature of each individual task they had worked on. They provided this information through a simple web interface described in Chapter 11.

Using the web interface, the participants recorded very basic information about themselves, including their age, gender, country of origin, and information about their educational achievement, their present job title, and their level of experience in that role. They were asked to categorize, in very broad terms, the type of software project they were working on at the time of logging their work.

The participant could enter a job description (unit tasks) along with some associated information regarding the job description. The associated information included duration, priority, status, interaction and any commentads/notes on the job. The job description field contained some options such as test planning, test execution, error reporting, test infrastructure building, testing tool maintenance and so on. These options were prepared from a review of software testing related job descriptions. We reviewed job descriptions in a number of job advertisements and listed the unit tasks presented as options for job descriptions. The participants could also write down a job description if the available options were not appropriate. The job descriptions usually were small blocks of text describing the unit tasks of the tester role.

Participants were encouraged to log as they work, or at least, at the end of each working day. There was also an option to edit already submitted jobs.

7.2.2 Survey of Software Testing Job Advertisements

In this approach, we collected the job descriptions of software testers from popular job web site monster.com [116] over a period of five days. We used an automated script written using PHP for this purpose. The URL containing the list of jobs resulted searching with "software testing" in the job website was given as input to the script. The script then collected the URLs to the individual job advertisements and parsed the job descriptions from those URLs. The collected job descriptions for testers were then analysed manually and unique job descriptions were listed.

7.2.3 Survey of Bug Descriptions in Bug Repositories

Many organizations employ bug repositories to keep track of the bugs of a software project. These repositories are hosted on local server of the organization and the access to these repositories is restricted and not permitted for public use. However, there are a number of open source bug repositories that contain same type of information. So we considered open source bug repositories in this method.

There are number of open source bug repositories, and as such the number of available bug reports is huge. To draw a boundary of our search space we limited our search to bug reports of testing related products of Eclipse and Firefox repositories. The products were Eclipse Java Development Tool (JDT) and Firefox (Server software) Testopia. A brief description about the products is given below:

Eclipse Java Development Tool (JDT)

JDT is a plug-in for the Eclipse software [117]. This is a Java IDE that helps development of Java applications using Eclipse. The JDT plug-ins provide APIs so that they can be extended by other tool builders. It has some core plugins - JDT APT, JDT Core, JDT Debug, JDT Text, JDT UI. JDT APT adds annotation processing support to Java projects in Eclipse. JDT core defines the non-UI infrastructure and provides code assist and code select support. JDT Debug supports debugging. JDT Text provides Java editor. JDT UI supports package explorer.

Firefox Testopia:

Testopia is a test case management tool. This tool is used for tracking test cases, integrating bug report with test case run results [118]. It is used for both a test case repository and management system and meets the needs of small to large size software testing organisation. It was developed to provide a central storehouse for the collaborative works of software tester.

The tool can be used as an extension to the open source issue tracking software- Bugzilla.

Rationale for selecting the tools

Generally there is no deliberate software testing team for most of the open source projects. The bugs of these projects mostly are reported by the users of the projects. Eclipse is an editor mostly used by JAVA programmers. On the other hand the primary users of Testopia are testers. If most of the users are not deliberately testing the tools then we can assume they report the bugs that they encounter while using the tools for development of other software.

Both these tools can be used during the testing phase of a software development life cycle. Thus the bug reports related to these tools may contain useful information about the testing of the software under development.

Procedure

We used an automated script written in PHP to search for all bugs that contained the keyword “test” in the summary and/or in comments fields of the bug reports. The bug ID of the resultant list of bugs was stored in a file. Another script, also written in PHP, read the bug IDs from the file and parsed the summary, description, comments, reporting time, reporter information of the respective bugs. The extracted information was then saved in a database stored on the researcher’s computer. The information of a subset of the resultant list of bugs were analysed manually. We found that the “steps to reproduce” section in the description of the bugs contained the information of the tasks performed to generate the bugs. We prepared another script to parse the “steps to reproduce” section from the description and the comments of all the resultant list of bugs. We discarded the bugs from the list that did not contain any information on “steps to reproduce”. These “steps to reproduce” of all the bugs were then analysed manually to identify distinct software testing tasks.

7.2.4 Analysis

The nature of the data collected in this research was mostly qualitative. Each of the methods adopted for this research resulted in a large textual dataset. We adopted the following steps to analyse the collected data:

1. Unit tasks were identified and listed from the complete worklogs, job descriptions and bug descriptions.
2. The frequency of each unit task was counted.
3. The listed unit tasks were analysed to group similar tasks in common categories.

7.3 Results

7.3.1 Case Study of Software Testers' Worklogs

There were a total of 6 cases (paticipants), from Australia, New Zealand, Germany and Bangladesh. Of these, two were female. Three of the cases were from 26-30 years age group.

Three cases mentioned their main job responsibility was to "test modules/programs developed by others". Two cases noted their main job responsibility was "managing testers within a project". One case mentioned main job responsibility was "testing and managing". Two cases had more than 5 years of experience and two cases had between 3 to 5 years of experience. The other two cases had 1 to 3 years of experience and less than a year of experience respectively.

All of the cases worked in different companies having between 35 to 40000 employees. Two of the cases mentioned they follow an “agile” development process while two other cases said they use “waterfall and agile” development process. “Waterfall” and “Partially agile” was followed by one case each. We also asked the cases to note the application domain they work in. They could choose more than one option in this category. Four cases mentioned their application domain was “Web development”, three cases worked in “Desktop application development”, two cases worked in “Mobile application development” and one case worked in “Backend application development”.

The demographic information of the cases is summarized in Table 7.2. To ensure anonymity, we denote the cases with A, B, C, D, E and F.

Table 7.2: Research summary

Case	Gender	Age	Nationality	Main job responsibility	Experience	Number of employee	Application domain	Development process	Project status
A	Female	51-60	Australia	1. Testing programs developed by others 2. Managing testers within a project	More than 5 years	200	Web development	Waterfall	Maintenance

B	Male	26-30	Bangladesh	Testing programs developed by others	More than 5 years	35	Web, desktop application, mobile application and backend application development	Agile	Running
C	Male	31-40	Germany	managing testers within a project	3-5 years	7000	Desktop application development	Agile	Some months before going live

D	Female	41-50	Germany	managing testers within a project	1-3 years	40000	Desktop application development	Waterfall and agile	Different projects at different status
E	Male	26-30	New Zealand	testing modules developed by others	1 year	6000	Web and mobile application development	Waterfall and agile	series of projects at multiple phases
F	Male	26-30	Australia	testing modules developed by others	3-5 years	300	Web development	Agile	Close to going live

Case A

Case A was a senior tester as well as test manager. She had more than 5 years of experience. The project she was involved in during logging her work was at maintenance phase. The project was developed using waterfall development process.

She submitted worklog for 37 working days. Working days are counted excluding the weekends during the period of logging the work. The jobs frequently submitted by her includes (in order of frequency of occurrence)- "analysing requirement and functional design", "developing test suit", "maintaining test infrastructure", "debugging", "executing tests". A close review of her worklog revealed that she was involved in variety of jobs such as developing and executing test suits, reviewing others work, performing human resource responsibilities, supporting support staff, providing quote for testing and so on.

Case B

Case B was a tester having more than 5 years of experience in this role. The projects he worked in followed agile development methodology.

He submitted worklogs for 5 working days. From an analysis of his worklogs, 7 unique tasks were found. These are: (in order of frequency of occurrence) "Execute tests", "debugging", "upgrade test tools", "analysing requirements", "reporting test results", "helping other testers" and "test team meeting".

Case B spent most of his time executing tests. He indicated executing tests was his routine work. He did not indicate with whom he worked for the submitted worklog.

Case C

Case C was a manager of testers with 3 to 5 years of experience. He was working in an agile development process. He recorded his worklog over a few months before the project was live.

He logged his work for 10 working days. The unique tasks identified from his worklog include: “planning for resources and upcoming test phases”, “coordinating testing team”, “meeting”, “preparation work for upcoming test phases” and “executing test”. While coordinating testing team he worked with other testers. For planning for upcoming test phases he worked with the project manager.

Most of Case B’s unit tasks were related to planning and coordinating. These unit tasks were routine work. He indicated project team meetings were often interrupted works. An interrupted work was defined as the work carried out interrupting other work in progress.

Case D

Case D was also a manager of testers with 1 to 3 years of experience. She worked in a large company having 40,000 employees. She was involved in multiple projects with different status. The projects she worked in followed both waterfall and agile development processes.

The unique responsibilities identified from her worklogs include: “planning test orders for next financial year”, “checking accounting for system test efforts”, “coordinating test team”, “helping other testers”, “project team meeting” and “maintaining test infrastructure”. While helping other testers she worked with offshore testing partners. For planning for the next financial year and for coordinating testing team she worked with the project manager. She did the maintenance work with other testers of the team.

Case E

Case E was a tester with 1 year experience. He was involved in multiple projects running at different stages. The projects he was involved in followed both agile and waterfall development processes.

He submitted worklogs for 3 working days only. We found 3 unique tasks from her worklogs. These include: (in order of frequency of occurrence) “searching for, analysing and finalizing testing tools”, “requirement analysis” and “test team meeting”. She worked together with the project team for searching for, analysing and finalizing testing tools. Most of her tasks were routine work. He noted requirement analysis was interrupted work.

Case F

Case F was a tester with 3 to 5 years of experience in this role. He was involved in agile development process and he logged his work close to product release deadline.

He submitted his worklogs for 10 working days. Around 12 unique tasks were found from his logs. Among these responsibilities, “executing tests”, “reporting test results”, “explaining test results”, “retesting after defects were fixed”, “project team meetings”, “ maintaining test infrastructure”, “developing test suite”, “gaining domain knowledge” were prominent. He noted that retesting was a different responsibility since he did not execute regular tests while retesting after defects were fixed.

A close observation of his worklogs revealed that he spent a significant amount of time explaining test results after reporting them. He worked with developer, project team and project manager while reporting results, however, most of the time he worked with project manager for explaining test results. He worked with subject matter experts to gain domain knowledge. Reporting and explaining results were sometimes routine responsibilities, sometimes interrupted responsibilities. He also noted project team meetings were most of the time an interrupted responsibility.

Summary of Case Study A total of 6 studied cases submitted their worklogs for different time span ranging from 2 to 37 working days. We used SQL (SELECT DISTINCT(‘jobDescription‘) FROM ‘workDetails‘ and SELECT DISTINCT(‘jobDescriptionOther‘) FROM ‘workDetails‘)to find out unique strings from the submitted worklogs by all. The SQL returned 63 unique strings from the total submitted jobs. A manual inspection of those 63 job descriptions revealed that similar unit tasks were listed multiple times due to minor difference in description or spelling. For example “Peer review work done by another test team member” and “Peer review work done by other tester” was listed separately due to the difference in description, however these two tasks are actually same.

We eliminated the redundant responsibilities from the list and obtained a list of 44 unique tasks. These 44 unique tasks were analysed carefully and by grouping similar jobs together we found 12 distinct categories of jobs were performed by the cases. The 12 categories along with the unique tasks in each category are listed in Table 7.3. The last column of the table represents the frequency of unique tasks in each category.

Table 7.3: Summary of case study

Category of tasks	Unique tasks	Frequency of occurrence

Planning	Resource planning	6
	Planning upcoming test phases	1
	Planning test orders for next financial year	1
Research and Development	Looking for test automation tools that will meet our requirements	1
	Analysing requirement and functional design	38
	Gaining domain Knowledge	2
	Finalizing proof of concept on automated tools	2
	Consulting with Subject Matter Experts	1
	Discussion of change in direction for another project - proposal to provide system testing support	1
Review	Peer Review work done by another tester	13
	Human Resources Responsibility - annual review of tester on other project	2
	Review defects logged by other testers	2
	Document reviews	1
Testing	Setting up test environment	4
	Develop test specification	7
	Develop test suite	35
	Execute tests	29
Retesting	Retesting of fixed defects (not execution of regular test)	5
	Defect retest	1
Reporting	Reporting test results	7
	Explaining test results	5

	Turn Around Report - status of all tasks currently with test team	5
Debugging	Debugging	22
Main tenance	Maintaining test infrastructure	25
	Upgrade test tools, versions, environments	2
Managerial	Coordinating testing team	15
Adminis trative	Prepare and provide quote for testing (preparation and execution) for a change request	4
	Preparation work for upcoming test phases	1
	Checking accounting for system test efforts	1
	Admin tasks related to Test Lead	1
	Various administrative tasks	2
Meeting	Test team meetings	4
	Project team meetings	17
	Development Practice Meeting	1
	Company meeting	4
	Meeting with external stakeholder	2
Help and Support	Helping other testers	15
	Help developers with existing functionality	1
	Support for support team	4
	Provide support to trainers	1
Miscel laneous	Waiting for development to fix defects	1

	Work for company I work for (as distinct from company where I am contracted to)	5
	Work for Test Practice Group	1
	On Leave	1
	Total	299

Planning

The first identified category of software testing unit tasks from the worklogs is planning. There were two main types of planning related responsibilities: "planning for resource allocation" and "planning for future testing". One issue observed from the submitted worklogs is that planning-related unique tasks were mostly carried out by managers. This is not surprising since planning generally is a part of managing.

Research and Development

From the analysis of the worklogs we see that software testers perform a number of research and development type jobs. Most prominent of those are: requirement and functional design analysis. This task refers to transforming the identified functional, performance, interface and other requirements in to coherent descriptions of system functions. This tasks is, in fact reported maximum number of times by all the cases. Requirement and functional design is analysed by both testers and managers. Other tasks of this category include: searching and analysing testing tools, gaining domain knowledge and so on.

Review

A good proportion of software testers' time is spent in reviewing jobs done by other testers. They also review the defects reported by other testers. Cases reported document review as a unique task, however the type of document is not clear from this description. Conducting annual review was reported by a manager.

Testing

This category represents testing specific jobs such as setting up test environment, prepare test specification, developing test suit and executing tests. Developing test suite and executing tests are reported many times by the cases. A noticeable fact is that executing tests is reported by both testers and managers.

Retesting

This category also includes testing-related jobs. However, cases indicated that retesting of fixed defects did not always involve execution of regular test suite. This is why a separate category of jobs is formed. Another job of this category says defect retest. It is a bit unclear what cases wanted to illustrate with this. Our impression is that defects reported by customers or other stakeholders are retested and confirmed by the cases.

Reporting

This category includes reporting as well as explaining test results. From the worklogs we find that our cases also needed to report the status of testing.

Debugging

Debugging refers to the activity of finding the root cause of a defect. This is a distinct responsibility and hence is grouped separately. Debugging is reported number of times by managers and testers.

Maintenance

Both managers and testers spent effort for maintaining test infrastructure. This category also included up gradation of test tools and others.

Managerial

The most popular managerial job, as was reported by the managers, was coordinating their test teams. Other managerial jobs included preparation for upcoming tests, checking accounts and preparing quote for testing.

Administrative

Our cases reported they did a number of administrative jobs as well. Some of those administrative tasks were reported to be related to leading test team as well. Most administrative jobs were not specifically described in detail.

Meeting

Our cases participated in several different types of meetings. Most frequent of those were project team meetings. They also participated in test team meetings, company meetings, amongst others. No client meeting was specifically mentioned.

Help and support of others

Our cases helped other software testers in their work. They also provided support to the support team, mentioned differently, for example answering queries from the support team and helping support staff in analysing, replicating and finding defects. One case also mentioned about helping developers with existing functionality.

Miscellaneous

The jobs that could not be grouped with others, are organised in this group. These include waiting for developers to fix defects, working for a separate company (types of work is not described) and so on.

Whom cases work with?

For most of the tasks reported by our cases, they worked alone. However, for 10.03% of total submitted tasks they worked with other testers. They also mentioned working with (in order of frequency of occurrence) developers, business analysts, project managers, subject matter experts, delivery managers and so on.

7.3.2 Survey of Software Testing Job Advertisements

We found 47 job advertisements for software testers on the monster.com over 5 days. We listed the tasks illustrated in these job advertisements. A manual inspection was performed to determine the unique tasks from the list. We found 39 unique tasks. By grouping similar jobs together in the same fashion we applied in the case study reported earlier, we found 11 distinct categories of job. The categories with the unique tasks are reported in Table 7.4.

Table 7.4: Summary of job descriptions

Category of tasks	Unique tasks	Frequency of occurrence
Testing	Prepare test plans	35
	Develop test specification	3
	Write test cases	33
	Create test input	4
	Execute test cases	43
	Implement and manage automated testing package	40
	Retest after defects are fixed	4

Research and Development	Develop and maintain quality standards	20
	Gain knowledge of product	6
	Requirement analysis	12
	Suggest enhancements to test mechanisms	2
	Identify issues during new product development	1
	Learn new technologies	1
	Participate in process improvement	11
	Determine resource needs	1
	Design new test strategy	9
Writing and Reporting	Report test results	24
	Analyse testing report	16
	Track, record and report testing status	13
	Write test documents	10
Debugging	Determination of the cause of defect	3
	Find and recommend options to resolve testing issues	11
Assessment and evaluation	Assess code coverage	1
	Evaluate usability	2
Planning	Prepare and ensure timeline deliverables	11
	Participate in budget process	1
Managerial	Manage test team	4
	Setup and maintain test infrastructure	11
Supervision	Review others' work	21
	Provide training	8

Adminis trative	Perform software configuration	6
	Defect tracking	6
	Administrative work	7
	QA Review	2
	Release management	3
	Track and control change management process	2
Collabo ration	collaborate with other stack holders	14
	Participate in status meeting	3
	Communication with Clients	1
	Total	305

Testing

Most of the jobs identified from the job advertisements fall in this category. Testing specific jobs are grouped in this category. These include planning tests, writing test cases, preparing test input, writing automated testing programs, executing tests and so on.

Research and development

Recruiters expected that recruited testers will be involved in process improvements, and will maintain quality standards. They also expected that testers will learn new technologies and gain knowledge of their domain. These jobs along with requirement analysis, suggesting new enhancements to test mechanisms are grouped in research and development type of tasks.

Writing and reporting

From the analysis of job advertisements, we found that recruiters not only expect testers to write test reports. They design more writing jobs such as writing test documents, reporting test status for testers. Testers also seemed to be analysing test reports (most probably reports from others).

Debugging

In practice recruiters seemed to be expecting testers to perform debugging activities as well, including finding the root cause of defects and providing suggestions for correction.

Assessment and evaluation

As part of testing testers are expected to prepare metrics such as assessing code coverage and usability.

Managerial

Few job advertisements contained managing test team in the job responsibility description. As seen from our worklogs analysis, managing a test team, in practice, should include a number of unique tasks. However, the job descriptions lacked the detailed steps of managing a test team. So we listed managing test team as a unique task and grouped it with setting up and maintaining test infrastructure under managerial category of jobs.

Planning

Some recruiters expected testers would participate in budget preparation and timeline deliverable planning process.

Supervision

Some recruiter designed the role of the testers providing training to other testers as well as to supervising their work.

Administrative

This category of jobs contained a number of different unique tasks. Jobs such as defect tracking, track and control change management, release management are considered administrative jobs of a testers for this analysis. The category also included QA review and software configuration related jobs.

Collaboration

Similar issues to the managerial category of jobs were found in this category of jobs. A number of advertisements contained a requirement for collaborative work with other stakeholders. However, the nature of the collaboration and the specific type of stakeholder were not obvious from the job descriptions most of the times. Jobs such as participating in meetings (not specific type of meeting) and maintaining communication with clients were also considered collaborative type of jobs.

7.3.3 Survey of Bug Descriptions in Bug Repositories

The steps to reproduce a bug as described in bug reports were very specific job information regarding testing activities. We identified 24 unique tasks from the bug reports. Most of the jobs identified from these texts were testing related. We grouped most of the jobs in a broad category called "Testing". The rest of the jobs were put in "Reporting", "research and development" and planning category.

The jobs in the "Testing" category were subdivided in small groups. Management of test plans was put in the testing category (as different to the other two methods described in Section 7.3.1 and 7.3.2). The reason is that unique tasks, such as prepare test plan, cloning test plan and so on, were specific plans for a test. Firefox Testopia tool enables users to create plan for a specific test and then associate test cases to the plan. Hence the test plan related jobs are kept in testing category, different from what we did in worklog analysis, where test planning represented the planning activities of overall testing.

The jobs identified from the repositories are reported in Table 7.5.

Table 7.5: Summary of bug descriptions

Category	Groups	Tasks	Frequency of occurrence	
			Eclipse JDT	Firefox Testopia
Testing	Preparation	Preparing environment	9	33
	Test plan management	Writing test plan	0	16
		Modify testplan	0	3
		Searching test plan	0	13
		Exporting/cloning test plan	0	7
		Deleting test plan	0	2
	Test case management	Creating new test-cases	3	29
		Modify existing test-case	1	10
		Searching/selecting existing testcase	5	18
		Cloning test case	0	7
	Test run management	Create test run	0	10
		Edit test run	0	1
		Selecting test run	0	10
		Cloning test run	0	2

	Association of components	Linking test plan with test case	0	5
		Linking test case with test run	0	4
	Test execution	Executing test	5	4
		Creating programs for executing tests	1	0
		Check output	3	17
	Other	Change test status	0	8
		Write notes	0	1
Reporting		Create test report	0	7
Research and development		Read help files	0	2
Planning		Time estimation	0	1
Total			27	210

The list of unique tasks collected from the bug descriptions represent the part of testing jobs performed during using tools such as Eclipse JDT and Firefox Testopia. The jobs do not represent the overall role of a tester, however sheds some light on the detailed tasks that make up.

7.4 Threats to Validity

There are number of threats that can attenuate the validity of our findings. We discuss some of those here.

The collection of information from the worklog, job advertisements and bug descriptions for a period of time may not guarantee all practiced responsibilities of software testing are present in the information. Some responsibilities may fall outside the time frame of the period worklog was collected for, may not be mentioned in the collected job advertisements and bug descriptions. This is a common threat of this type of research. To overcome the threat we need to collect worklog, job advertisement and bug descriptions for infinite time which is impossible. However, to minimize the effect of this threat our plan was to collect as much data as feasible.

The information we are seeking to extract from this research is highly dependent on the organization, type of software being developed and the development process being followed. We tried to capture this information in collecting worklog by asking type of development process followed and number of employees in the organization, and business domain of the organization. However, in other two methods it was infeasible to obtains this information. Thus our finding do not ensure the same tasks are practiced in organizations operating in different business domains and using different development processes.

The type of data collected was qualitative. The qualitative data analysis process followed in this study required reading the bundle of text collected as worklogs and extracted from job advertisements and bug descriptions and listing the unique tasks. A lot of the time the analysis process required reading between the lines and interpreting the hidden meaning of the text. This needed a good understanding of the language. The initial analysis was conducted by the student researcher who is a non native English speaker. This presented the threat of interpreting the information incorrectly and listing wrong unique tasks. To overcome this threat part of the analysis performed was checked by the supervisors who were native English speakers and the student researcher was guided accordingly.

7.5 Discussion

From our research, we found that “Test” specific responsibilities of a software testers comprises of a number of unit tasks such as creating test cases, preparing environment, executing test cases and so on. These unit tasks were found from all three methods followed in this research study.

As a consequence of testing testers need to report results. We observed that testers also need to explain and analyse the test reports. They also perform other writing tasks such as preparing test documents, writing test status reports. The amount of writing and reporting related tasks performed by the testers indicate that they need good communication and writing skill. This also suggests that the development of better methodologies and tools for test reporting is potentially very useful. Along with testing specific knowledge, software testing related certification syllabus and undergraduate IT curricula should thus also include knowledge and practice to develop these skills.

We also found that software testers are engaged in many research and development group of tasks. Most frequent of this type of jobs evident from the worklogs is requirements analysis. This job was frequently mentioned in job advertisements as well. In addition to that testers also gain domain knowledge and conduct research about testing tools and new technologies. These are encouraged by the recruiters as well. Testers also participate in preparing quality standards and in improving process. In our opinion, these responsibilities seek the tester to be quite a good reader, possessing qualities such as attention to details, being well organized and so on. Some of these qualities were also listed by our participants in our preliminary survey as key factors influencing testers' performance.

Software testing tasks also include debugging tasks, such as finding the root cause of the defect, exploring and suggesting possible resolutions. This was evident from both the worklog and the job advertisements. To find the cause and the possible resolution of defects testers need to review the code. According to the study performed by Cunha and Greathead [31], Intutitive-Thinking types of people (as assessed with MBTI personality assessment test) are good at code review tasks. NT people are described to be "logical and ingenious" by the MBTI type definition [31]. This category of people is always "looking at the possibilities, theoretical relationships, and abstract patterns" [119].

Along with the testing and debugging responsibilities testers (mostly senior level such as managers) perform some managerial, supervision and planning related jobs.

Testers collaborate with other stake holders, such as programmers, managers and clients, and participate in meetings. The types of meetings were not clear from the job advertisements. However from a review of the collected worklog we see they participate in different type of meetings such as project team meeting, test team meeting, company meetings and so on. This indicates a significant amount of testers' time is allocated to meetings. Hence a different category called meeting is formed from the analysis of the worklogs.

7.6 Summary

This research study was an attempt to prepare a refined list of software testing unit tasks. We adopted different research methods to collect information from different available sources. From those sources of information we could prepare a list of unit tasks performed for testing. We also found that along with testing many other unit tasks such as debugging, planning, maintenance, managerial and collaboration with others are performed by testers.

We believe the identified category of jobs along with specific unit tasks of each category will help recruiters to design job responsibilities for testers. This will also help young graduates get better understanding of the responsibilities of this role in selection of their career choices. This will also benefit our research finding suitable human characteristics of software testers. In addition, some potential enhancements to tester training and education were identified including an emphasis on soft skills.

Chapter 8

Personality Traits of Software Developers

8.1 Introduction

Software testing is a distinct role from other software development roles. The influence of certain personal characteristics on this role is evident from our preliminary research reported in previous chapters. Different personality type is potentially one of these. The research study reported in this chapter addresses "RQ3: What personality traits are over-represented among testers?" and "RQ3.1: Are these traits different from other practitioners?" by investigating whether any particular personality trait is over represented among professional software testers and whether that is distinct from those of others involved in software development.

8.2 Methodology

In this research study, we collected the personality profiles of software engineers with the help of a survey. We used a web based survey as our research strategy since such a survey enables us to collect personality profiles of wide range of software engineers in a very short time. The survey was designed according to the six steps suggested by Kitchenham and Pfleeger [103]. The steps are discussed in the following subsections.

8.2.1 Setting the Objectives

The research study involved a group of software testers and a group of software developers who were involved in other roles of software development excepting software testing. The objective of the research study was to collect the personality profiles of the two groups and to conduct a comparative analysis to find out if there are any notable trends and significant differences among the two groups. The personality profiles of the participants were prepared based on the "Big five factor" model [120] of personality.

In order to find the differences, if any, of five major personality traits described in big five factor model, between software testers and the non testers, we assume the following alternative hypothesis:

H_A: There is difference in mean on the five personality traits (Neuroticism, Extraversion, Openness to Experience, Agreeableness and Conscientiousness) between software testers and the non testers.

In contrast to the alternative hypothesis we propose the following null hypothesis:

H_O: There is no difference in mean on the five personality traits (Neuroticism, Extraversion, Openness to Experience, Agreeableness and Conscientiousness) between software testers and the non testers.

8.2.2 Survey Design

We conducted a web based survey with a self-administered personality assessment questionnaire. One of the main benefits of web based surveys is that the responses are collected in an automatic fashion and participants can complete the survey questionnaire at their convenience. A potential disadvantage of web based survey, however, is the generalization of the sample. Due to the necessity of internet access sample is often skewed to the participants with internet access. In our survey, our intended participants were software engineers who generally would have internet access.

8.2.3 Development of Survey Instrument

According to Kitchenham and Pfleeger [103] researchers should search for relevant literature before developing a survey instrument at this stage. This has two key benefits. The first benefit is this will reduce the chance of duplicate research. The second is that the researchers can learn and, if possible, can adopt questions and experimental design from existing relevant research.

A similar research study to ours was conducted by Capretz [46] and reported the personality profiles of software engineers. However, software engineers were not divided into sub groups depending on their specific roles (e.g. programmers, analysts, software testers) for analysis. The personality of those software engineers was also assessed using the MBTI personality assessment test that identifies one of possible 16 “type” of personality. This is different compared to the contemporary trait based assessment of personality.

We used the 50 item IPIP personality assessment test [90] for this research study. This test is designed based on the “Big five factor” model [120] of personality, which is the most commonly used contemporary model of personality psychology. The rationale behind selecting this personality test is that it can be used royalty-free and the test items as well as the scoring rules are available. The shorter version of the test could also be completed very quickly.

We used a custom-built data collection website for this survey described in Chapter 11.

8.2.4 Questionnaire Design

The survey was divided into two sections. The first section collected general demographic information of the participants along with their role in software development. The second section contained the IPIP NEO personality test items. The responses to each item could range from “Very inaccurate” to “Very accurate”.

8.2.5 Evaluation of Survey Instrument

A pilot survey was administered to validate the survey instrument. In the pilot survey the URL of the survey website was sent to some PhD students of Faculty of Information and Technology in Swinburne University of Technology. Based on the feedback obtained from the pilot survey minor spellings and a duplicate item were corrected.

8.2.6 Obtaining Valid Data

We used cluster and purposive sampling to recruit participants for this survey. In cluster sampling, instead of selecting individuals from the population randomly, clusters of individuals are selected and within one cluster all individuals are included in the sample [100]. In this survey we requested permission from the 12 LinkedIn and 12 Yahoo! groups that gave us permission in our preliminary survey [121; 122]. 5 Yahoo! and 3 LinkedIn groups approved us. In this process we could select clusters of participants who are associated with software testing-related mailing lists and groups. In comparison to sending invitations to individuals, sending email to the groups (clusters from the population) enabled us to invite larger group of participants in a shorter time.

In the purposive sampling process the sample is "hand picked" on the basis of relevance and knowledge [100]. As part of this process we attended a software testing related industry conference in Australia and posted to an industry conference email list related to software development. We also tweeted on the Twitter feed of the developer conference with the help of the organisers. Given the maximum size of a Twitter message is 140 characters, we sent out a "teaser" message to encourage those who read the message to investigate further. Our Tweet was: "Are developers from Mars and testers from Venus? Help Swinburne university researchers find out: <link of the website>".

The participants could register for a draw of two $100 Amazon.com gift vouchers by providing their email addresses. The email addresses were stored in a separate database table and were not associated with their responses.

8.2.7 Data Analysis

We report results with descriptive statistics. We used Mann-Whitney U test to find the significance of difference of mean scores on five major personality traits. We used Hedge's g to find the effect size to quantify the difference in mean. We also used power analysis to determine the statistical significance of our findings.

8.3 Results

8.3.1 Demographic Information

Total 182 software engineers participated in the survey among them 45.1% were software testers. Among the rest of the participants 57% were programmers and 28% were managers. There were also business analysts, consultants, Architects, Software Product Designers and so on. The Gender and nationality of the participants are given in Table 8.1.

Table 8.1: Gender and nationality of the participants

Criteria	Software testers (%)	Non testers (%)
Gender		
Female	16.5	12.1
Male	28.6	42.9
Nationality		
Australia	7.7	14.3
Bangladesh	8.2	8.2
Brazil	0	0.5
Canada	1.1	2.2
Croatia	0.5	0
Ecuador	0.5	0
Egypt	0.5	0
Germany	1.6	1.1
Hungary	0	0.5
India	4.9	2.7
Indonesia	0.5	0
Iran	0.5	0
Israel	0.5	0
Malaysia	0.5	0.5
Nepal	0.5	0
New Zealand	1.1	1.1
Pakistan	2.7	0
Peru	0.5	0
Philippines	0.5	0
Romania	1.1	1.6
Serbia	0	0.5
South Africa	0	0.5
Spain	0	1.1
Turkey	0	0.5
Ukraine	0.5	0
United Kingdom	2.7	2.7
United States of America	6	13.2
Not selected	1.6	3.3

The majority of our participants were male irrespective of their role. This is not surprising since the majority of practitioners in the IT field are male [123]. However, a noticeable fact is that, there were more female participants in the software testers group than in the non testing group. The highest number of participants selected their country of work as "Australia". The second highest number of participants was from "United States of America".

The type of employment of the participants and their experience are shown in Figure 8.1 and 8.2, respectively.

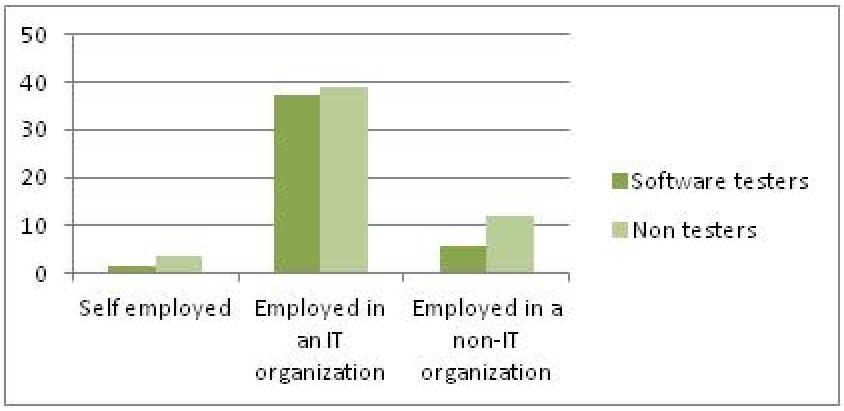


Figure 8.1: Type of employment

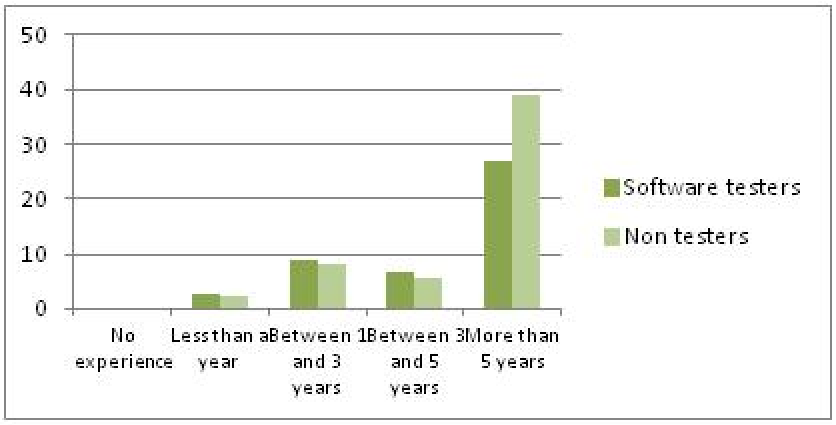


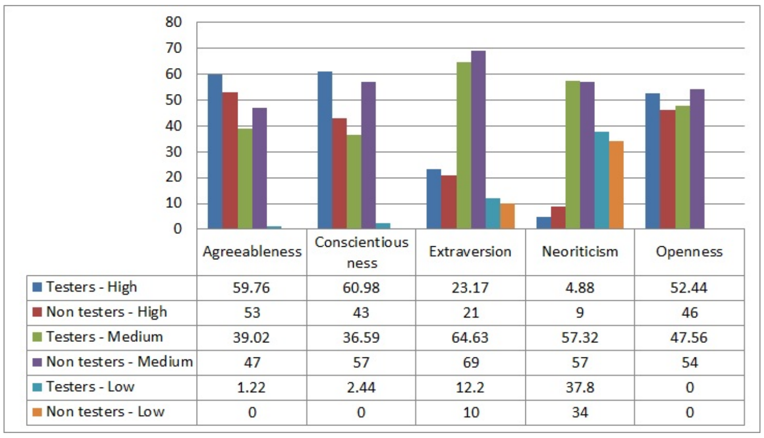
Figure 8.2: Experience

From Figure 8.1, we find that most of our participants were employed in IT companies. A small portion of the participants worked in non IT organizations and a very small portion of them were self employed. Figure 8.2 shows experience of our participants. Majority of our participants had more than 5 years of experience

8.3.2 Personality Distribution

Figure 8.3 shows the percentages of participants with different levels of five major personality traits obtained in our study.

The numerical score on the five major personality traits were categorized in three distinct levels- low, high and medium, suggested by Johnson [124] and applied by Norsaremah et al. [125] on New Zealand based student sample. According to the scheme, if the score lies within lowest 30% boundary, the level is low, if the score lies within middle 40% the level is medium and if the score lies within the highest 30% then the level is high.



	Agreeableness	Conscientious ness	Extraversion	Neoriticism	Openness
🟦 Testers - High	59.76	60.98	23.17	4.88	52.44
🟥 Non testers - High	53	43	21	9	46
🟩 Testers - Medium	39.02	36.59	64.63	57.32	47.56
🟪 Non testers - Medium	47	57	69	57	54
🟦 Testers - Low	1.22	2.44	12.2	37.8	0
🟧 Non testers - Low	0	0	10	34	0

Figure 8.3: Personality distribution

From the distribution presented in Figure 8.3 we see there were a number of non-testers with medium conscientiousness compared to testers who were highly conscientious. Both testers and non testers were agreeable, extravert and open to an almost equal degree. However, we noticed a higher number of non-testers with high neuroticism compared to testers in this group.

8.3.3 Tests of Normality

We have applied the Kolomogorov-Smirnov and Shapiro Wilk test [104] to our sample, as shown in Table 8.2. For these tests of normality, if the Sig. value is less than 0.05 then the distribution significantly deviates from the normal distribution. From the obtained Sig. values via the Kolmogorv-Smirnov test, we see that, except for extraversion, the distribution of scores for other factors significantly deviated from the normal distribution. Using the Shapiro-Wilk test, we found the distribution of scores agreeableness, conscientiousness and neuroticism significantly deviated from normal distribution. From the results of both the tests we found only the distribution of extraversion did not significantly deviate from the normal distribution.

Table 8.2: Tests of Normality

	Kolmogorov-Smirnov a			Shapiro-Wilk		
	Statistic	df	Sig	Statistic	df	Sig
A	.092	182	.001	.977	182	.005
C	.067	182	.045	.984	182	.036
E	.062	182	.081	.987	182	.096
N	.069	182	.036	.976	182	.003
O	.086	182	.002	.987	182	.090

a. Lilliefors Significance Correction, N= Neoroticism, E= Extraversion, O= Openness to experience, A= Agreeableness, C= Conscientiousness

8.3.4 Internal Consistency

We have calculated the Cronbach's Alpha [126] to determine the internal consistency of the personality test items. Cronbach's Alpha is a measure of reliability of a test and can range from 0 to 1. The closer the Cronbach's Alpha is to 1 the greater the reliability is. The obtained Cronbach's Alpha of the items for agreeableness, conscientiousness, extraversion, neuroticism and openness to experience are reported in Table 8.3. We also present the Cronbach's Alpha of five major personality traits reported by Goldberg with original scales and reported by Buchanan et al. [127] with revised scale on an internet sample of 2448 people.

Table 8.3: Cronbach's Alpha

	Our sample	50 Item original scale by Goldberg	50 item revised scale on Internet sample by Buchanan
Agreeableness	0.74	0.77	0.76
Conscientiousness	0.81	0.81	0.84
Extraversion	0.85	0.86	0.88
Neoroticism	0.83	0.86	0.83
Openness to experience	0.67	0.82	0.74

According to the proposed interpretation of George and Mallery [128], on our sample the reliability for conscientiousness, extraversion and neuroticism were good, reliability for agreeableness was acceptable and reliability for openness to experience was questionable.

The internal consistency of our sample was similar to those reported by Buchanan et al. [127] on internet sample of 2448 participants. Except for neuroticism our internal consistencies were also similar with those reported by Goldberg with original scales.

8.3.5 Hypothesis Testing

The mean and standard deviation of each of the five major personality traits measured by the 50 item IPIP test on our sample is presented in Table 8.4. The scores on each factor could range from 10 to 50 inclusive. We applied Mann Whitney U test to find if the mean score on each factor significantly varied between software testers and the non testers. This test is applied to compare the differences of mean between two independent groups when the normaility of underlying distribution is questionnable [104]. The results of the 2 tailed Mann Whitney U test are given in column 8 of Table 8.4. We see only for conscientiousness that $p <= 0.01$, This indicates that the testers scored significantly higher in conscientiousness than non-testers. No other significant differences were found.

8.3.6 Effect Size

Statistical tests such as Mann Whitney U test help us to find of the mean of two independent groups differ significantly. However, these tests do not indicate the extent of differences. An effect size can help us to find the magnitude of mean differences. We have applied Hedge's g to find the effect size of the mean differences on the five major personality traits. The calculated effect size for conscientiousness is 0.39 that refers to a medium effect. A medium effect implies that the mean score on conscientiousness between software testers and non testers

Table 8.4: Population distribution

	Software testers		Non testers		Combined		Mann Whitney U test (Between software tester and non testers)	Effect size	Percentage of non testers who would be below average person in software testing group	Statistical power
Factors	Mean	SD	Mean	SD	Mean	SD	p			
A	38.51	5.53	37.55	5.21	37.98	5.36	0.191	0.18	58%	0.27
C	38.37	6.03	36.19	5.79	37.17	5.98	0.011	0.39	66%	0.76
E	32.91	7.62	31.51	7.24	32.14	7.43	0.155	0.19	58%	0.27
N	25.07	6.98	25.71	7.3	25.42	7.15	0.549	-0.09	54%	0.10
O	37.54	5.21	37.32	5.43	37.42	5.32	0.892	0.04	50%	00.5

N= Neoroticism, E= Extraversion, O= Openness to experience, A= Agreeableness, C= Conscientiousness

are likely to be different. An interpretation of the effect size is presented in Table 8.4. We see that approximately 65% of non-testers would be below average conscientiousness of a person in the testing group.

8.3.7 Power Analysis

We applied post-hoc power analysis to compute the probability that the null hypothesis was correctly rejected. The computed power of our hypothesis testing is reported in Table 8.4. We see the power ranged from 0.05 to 0.76. We see that for conscientiousness we obtained a high statistical power. For the other factors, such as Openness to experience, the statistical power was relatively low. This means that we can have confidence in our result for significant difference in conscientiousness between testers and non-testers, however less confidence for the other results.

8.3.8 Comparison with The General Population

The designers of the IPIP personality assessment test discourage establishment of and comparison against any norms. We were unable to find any norm on general population to compare our sample with. The only available results (mean and standard deviation) of application of the 50 item IPIP test on an internet sample of 2448 self-selected participants was reported by Buchanan [129] in an unpublished paper. However, the number of items for each factors was not same as has been used in this study. Therefore we have not directly compared our results with those reported by Buchanan.

Even though the test applied by Buchanan used different items, those measure the same five factors. Therefore we calculated the inter correlations in our sample and compared with those reported by Buchanan et al. [127] using the method proposed by Fisher [130]. We found the intercorrelations were higher on our sample, indicating that the personality scores of our sample were more homogenous than those of Buchanan et al [127].

8.4 Threats to Validity

There are a number of possible threats to the validity of our study.

8.4.1 Threat to Internal Validity

In web based surveys a major threat to the internal validity is the random responses of participants. The participants of this survey could register for a draw of 2 $100 Amazon gift vouchers that may be a motivating factor for participation. However, the interest and enthusiasm observed in inviting participants contradicts with this view. The fact that only 45.6% participants registered for the draw also illustrates that not every participant was motivated by the reward.

8.4.2 Threats to External Validity

One possible threat to the external validity of this research is generalization of the findings. This is a common threat to such research conducted with a limited sample. Since it is impossible to conduct the study with an entire population, the findings can always be questioned for any bias caused due to sampling. However, the demographic information provided in Table 8.1, shows our participants were from different countries. Most of those countries have well established software industry of their own. Thus we can consider our sample to be broadly representative.

8.4.3 Threat to Construct Validity

For this research study we have employed a freely available personality test instrument. The reliability of the test adopted for this research can pose a threat to the construct validity of the research. Although the personality assessment test is unrestricted, the test has become popular and is widely used [91].

8.5 Discussion

The research was designed to find the personality profiles of software testers and other professionals related to software development. Our aim was to observe the personality profiles and find if there is any difference between software testers and other IT professionals. From our perceptions and the findings of our earlier survey, we hypothesized that software testers would, as a group, exhibit differences in personality traits compared to others involved in software development.

The results indicate that there is no significant difference in mean scores on agreeableness, extraversion, neuroticism and openness to experience between software testers and the non-testers for our sample population. However we found significant difference in mean on conscientiousness. The effect size results indicate that the probability of software testers to be high on conscientiousness is higher than the non-testers.

The distribution of different personality levels presented in Figure 8.3 indicates that in our sample we observed software testers scored significantly higher on conscientiousness compared to the non-testers. Conscientiousness is related to personal organization of people [88]. Highly conscientious people tend to be more organized, disciplined and hard working. Higher number of software testers with higher conscientiousness conforms to the general assumption, found in our earlier informal survey of practitioners, that software testers generally need to be highly organized, disciplined and hard working.

The qualities exhibited by highly conscientious people seem to be important to succeed in any profession. However, these qualities might be particularly important for software testers. Capretz and Ahmed [14] analysed the job responsibilities of software testers as mentioned in job advertisements and suggested that sensing and judging type of people categorized with MBTI personality assessment test will be more successful as software testers. Sensing and judging types of MBTI are associated with conscientiousness of five major personality traits [131]. Our study's finding that software testers are more highly conscientious than others in the IT profession supports Capretz and Ahmed's perception.

Greathead and Cunha [31] found that intuitive and thinking type students assessed with MBTI were better at performing code review task. According to Furnham [131], Conscientiousness is associated with thinking and sensing type of MBTI. Sensing is opposite to intuitive type in the dichotomous scale of MBTI. The observation of Greathead and Cunha, thus partly supports the influence of conscientiousness on effectiveness of software testing tasks.

8.6 Summary

The purpose of this research was to find any difference in personality on five major traits between software testers and others involved in software development. We conducted a web based survey to collect personality profiles of nearly 200 IT practitioners. From the analysis of their personality profiles we found a significant difference on the conscientiousness trait between the software testers and other software developers. Software testers in our sample scored significantly higher than other software developers. This is indicative that software testers differ in their personality from others involved in software development. The difference in personality between software testers and others, makes our assumption more stronger that personality traits might influence the effectiveness of software testers. We investigate such influence in our next study.

Chapter 9

Influence of personality traits on software testing

9.1 Introduction

The results of our preliminary survey reported in Chapter 5 indicate that certain human factors were viewed as crucial to being an effective tester. Most of the factors, we believe are related to the personality traits. However, specific personality traits to be influential for testers could not be identified in the detailed survey. We designed a quasi experiment to investigate the relationship between the psychological aspects of individuals who test software, and their performance as testers. In the quasi experiment we attempted to answer "RQ4: Do personality traits influence software testing performance?" and "RQ4.1: If yes, which trait(s) has the maximum influence?" by examining the connection between the personality profile and testing performance of later-year software engineering students. This chapter describes the quasi experiment.

The effectiveness in software testing can be influenced by a number of factors. Although the quasi experiment was specially designed to investigate the effect of personality, we also recorded experience and background knowledge of testing of the participants and examined the effect of these factors on the effectiveness of testing. The background knowledge of software testing was determined by the information whether participants studied any unit on software testing.

9.2 Experiment Design

The quasi experiment was divided in two broad phases- Assessment of personality and Assessment of effectiveness in software testing. Following subsections describe about each of the phases and other details of the quasi experiment.

9.2.1 Assessment of Personality

MBTI has been commonly used in research relating personality and IT practitioners. However, neither the MBTI, nor Jungian personality theories, on which the axes of the MBTI are based, are widely used in modern personality psychology research. A major reason is because the MBTI does not come with norms based on continuous scores, restricting the scope of statistical analysis [37].

We have, instead, chosen to use the NEO personality inventory test (NEO PI-3), proposed by Costa and McCrae [132], and based on the five factor model of personality. In comparison to MBTI, NEO personality inventory tests are widely accepted by personality psychology community, and provide continuous normed scores on the standard five-factor model of personality. Tests based on big five factor model of personality such as NEO inventory of personality assessment has gained much popularity in recent times [48].

9.2.2 Assessment of Effectiveness in Software Testing

To our knowledge, there is no available method or test for assessment of effectiveness in software testing. We have used a faulty Java program to assess the effectiveness in software testing in this study. The program was developed as an assignment as part of "Programming in Java" unit in Swinburne University of Technology. The program calculates all possible loop free network topologies given the location of different towers. The locations are defined with latitude and longitude. The program also finds the shortest topology among all. A number of anonymous student assignments were collected and tested by the researchers. There were 18 different types of bugs encountered in those programs. We selected the student assignment program that had least number of bugs in it to be used for the testing task in this study. From the list of the bugs found in the assignment programs, 13 bugs along with 7 more bugs (which were not found in the student programs) were injected in the assignment program to be used for the study. Thus the testing task program had a total of 20 known bugs in it. The main reason behind injecting detected bugs is that the bugs represent real world bugs as they were created by the student programmers writing the program. The severity of the injected bugs was decided on the classification taken from [133]. This scheme is broadly accepted in many projects including Microsoft [134]. To ensure that the injected bugs represent real world bugs, we have compared the type of bugs with two well-known bug classification scheme Knuth's errors and Eisenstadt's bug war stories [135]. Table 9.1 presents the comparison along with the description of the bugs.

The program used for our testing task had total 1017 lines of code and the maximum cyclomatic complexity of the program was 7 (Calculated using Eclipse-Metrics plugin [136]). Figure 9.2.2 shows occurances of different cyclomatic complexity.

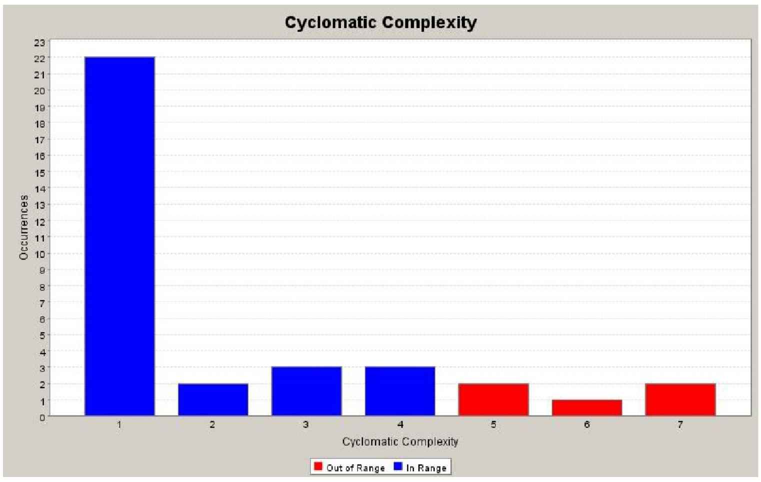


Figure 9.1: Occurances of different cyclomatic complexity

9.2.2.1 Assessment Metrics

There are a number of proposed metrics in the literature for assessing testing effectiveness [62], [17]. Three metrics were used for our purpose: Bug location rate,

Table 9.1: Description of Injected Bugs and Comparison with Knuth's Errors and Eisenstadt's Bug War Stories

Bug No.	Correct program behaviour	Incorrect program behaviour (Bug)	Fault description	Fault Source	Severity	Knuth's errors	Eisenstadt's bug war stories
1	The output is printed in the console and in the file	The output is printed only on the console	Missing functionality	Students	Low	forgotten function	unsolved
2	Pair of cities are printed with their names	Pairs are printed with numbers	Wrong implementation	Students	Medium	algorithm awry	
3	Prints right number of pairs possible among the given cities	Prints wrong number of pair of cities	Wrong implementation	Researchers	High	blunder or botch	
4	Prints right distance between the pair of cities	Prints wrong distance between pair of cities	Wrong function call	Researchers	High	mismatch between modules	des.logic
5	Prints pairs with right (name/number) of cities	Prints pairs with wrong (name/number) of cities		Researchers	High	blunder or botch	
6	Prints right number of topologies possible among the given cities	Prints wrong number of topologies among the given cities	Wrong index	Students	High	blunder or botch	lex/var
7	Prints right topology distance	Prints wrong topology distance	Wrong index	Students	High	blunder or botch	lex/var
8	Selection of right shortest distance	Selection of wrong shortest distance	Wrong parameter	Students	High	blunder or botch	var
9	Not printing duplicate topologies	Printing duplicate topology		Researchers	Medium	blunder or botch	
10	Right display of distance unit	Wrong display of distance unit	Wrong string literal	Researchers	Medium	trivial typo	
11	Should calculate distance for all values	Cannot calculate topologies for special value (gives exception with 2 cities)		Students	Critical	Forgotten function	lang
12	Calculates right distance for cities with <0 degree	Calculates wrong distance for cities with <0 degree	Missing Bracket	Students	High	trivial typo	lex
13	Should work with any number of cities	For more than 6 cities gives memory limit exceed exception	Memory	Students	High	Language liability	mem
14	Use of right conditional operator in line 286 wirlessNet.java	Use of wrong conditional operator in line 286 wirlessNet.java	Wrong implementation	Researchers	High	trivial typo	var
15	Use of right conditional operator in line 309 wirlessNet.java	Use of wrong conditional operator in line 309 wirlessNet.java	Wrong implementation	Researchers	High	trivial typo	var
16	Calculates right distance for cities with 0 degree	Calculates wrong distance for cities with 0 degree	Missing equal sign	Students	High	trivial typo	var
17	Should print error message when number of parameters mismatch	No output when number of parameters mismatch	Number of parameter mismatch	Students	Critical	surprising scenario	behav
18	Should print error message for null parameter value	No output for null parameter value	Null parameter	Students	Critical	surprising scenario	behav
19	Should print error message when parameter types mismatch	No output with parameter type mismatch	Parameter type mismatch	Students	Critical	surprising scenario	behav
20	Right method name "getTopologies"	Wrong method name "getTopology"	Wrong implementation	Students	Low	trivial typo	

Weighted fault density, Bug report quality. Given that there are no universally-accepted criteria for tester effectiveness, we judged that these three metrics were plausible and practical for use in our experiment.

- **Bug Location Rate:** This is measured as the number of bugs found in comparison to the time taken for testing [133].

  Bug Location Rate $= \frac{Number of bugs found}{Time taken for testing (in minutes)}$

  Bug location rate could range from 0 to infinite.

- **Weighted Fault Density:** The weighted fault density can be measured as:

  Weighted Fault Density $= \frac{(W_1 * S_1)+(W_2 * S_2)+(W_3 * S_3)+(W_4 * S_4)}{N}$

  where:

  W denotes assigned weights $W_1 > W_2 > W_3 > W_4$. S denotes severity of faults assigned according to [133]. N denotes number of total bugs found.

  There were 4 critical, 11 high, 3 medium and 2 low bugs in the software. For this experiment we used the following weights: $W_1 = 0.4, W_2 = 0.3, W_3 = 0.2$ and $W_4 = 0.1$. The total weighted fault density could range from 0 to 0.4.

- **Bug report quality:** This is assessed using the IEEE standard of Test Documentation [137]. According to the standard, there are 8 separate documents that should be given as part of a bug report. As our test sessions were small, as well as the provided software to test was also a small and simple one, we did not expect the participants to provide separate document(s) on each of the bugs found. We examined whether any information on each of the 8 standard deliverable fields were present for each of the bugs reported in the bug report prepared by a student and assigned a score from

0 to 1 for each of the field. Thus the total score of bug report quality could range from 0 to 8.

The total number of bugs found was calculated as the number of reported valid bugs (Total number of reported bug - total number of reported invalid/false bug).

9.2.2.2 Assessment of Overall Effectiveness

We did not find any indication of how to assess overall effectiveness using the above defined metrics in the testing literature. For assessment of overall effectiveness using these metrics we followed two possible approaches:

- **Total score:** In this approach we summed the score of bug location rate, weighted fault density and bug report quality of a participant to derive overall score. The score is denoted as O_{sum}.

  O_{sum} = bug location rate + weighted fault density + bug report quality

- **Weighted total score**: In this approach we assigned different weights to the assessment metrics according to their importance on total score to get an overall score. According to Kaner [64] the number of bugs found by a tester may be affected by a variety of factors. He emphasized qualitative assessment of bug report quality than accounting for bug counts [17]. In our previous study [121], we found that "Quality of bug report" is more important than "Severity of bugs found". Therefore, we assign more importance to bug report quality. The other two proposed metrics are based on bug count and we can not distinguish the varying importance of those. We assign the following weights to the three proposed metrics (bug location rate = 0.3, weighted fault density = 0.3 and bug report quality = 0.4) and sum the weighted metrics to get weighted total score, denoted as O_{wsum}.

O_{wsum} = (0.3 * bug location rate) + (0.3 * weighted fault density) + (0.4 * bug report quality)

9.2.2.3 Validation of The Instrument

After a pilot study conducted with 9 volunteer student participants, minor changes were made in the description of the specification document provided with the testing task.

9.2.3 Research Question

The formal research question that we intend to answer in this study, states:

Formal Research Question: Is there any association between personality traits assessed by five factor model of personality and effectiveness in software testing? Accompanying this formal research question we also expect to answer the following other research questions:

Other Research Question 1: Is there any significant difference in the effectiveness in software testing between the group of participants who have experience in software testing and those who do not.

Other Research Question 2: Is there any significant difference in effectiveness in software testing between the group of participants who studied software testing related unit and those who did not.

9.2.4 Variables

We consider the five factors of personality assessed by NEO PI-3 (Neuroticism, Extraversion, Openness to Experience, Agreeableness and Conscientiousness) as our independent variables and the two measures of overall effectiveness (O_{sum} and O_{wsum}) as our dependent variables. We also consider the three different measures of effectiveness (Bug location rate, weighted fault density and bug report quality) as our dependent variables for analysis. We also consider the experience (whether participants had experience in software testing) and background knowledge (whether they studied any software testing unit) as independent variable. The list of dependent and independent variables are given in Table 9.2.

Table 9.2: Variables

Independent variables	Dependent variables
Score on Neuroticism (N)	Total score (Osum)
Score on Extraversion (E)	Weighted total score (Owsum)
Score on Openness to Experience (O)	Bug location rate (BLR)
Score on Agreeableness (A)	Weighted fault density (WFD)
Score on Conscientiousness (C)	Bug report quality (BRQ)
Experience in software testing	
Background knowledge of software testing	

9.2.5 Hypothesis

In order to test the effect of five personality traits on the effectiveness of software testing we assume the following alternative hypothesis:

H_A:There is an association between different levels of five personality traits (Neuroticism, Extraversion, Openness to Experience, Agreeableness and Conscientiousness) and the effectiveness in software testing.

In contrast to the alternative hypothesis we propose the following null hypothesis:

H_O: There is no association between different levels of five personality traits (Neuroticism, Extraversion, Openness to Experience, Agreeableness and Conscientiousness) and the effectiveness in software testing.

Difference in levels of five personality traits can be calculated from the T scores on each personality factor assessed by NEO PI-3. The test suggests five levels depending on the T score. For ease of calculation we have combined the five levels to three levels: Low ($T <= 44$) , Average ($T = 45 to 55$) and High ($T >= 56$). Participants were grouped in either of the three levels according to their T score on each of the five factors for hypothesis testing.

In addressing the other research questions described in Section 9.2.3, we propose the following null hypotheses:

H_O_1: There is no significant difference in mean effectiveness between the group of participants who have experience in software testing and those who do not.

H_O_2: There is no significant difference in mean effectiveness between the *group of participants who studied software testing related unit and those who did not.*

9.2.6 Participants

The participants of the study were students of FICT of Swinburne University of Technology and Monash University. The researchers gave a small 5 minute talk at the end of the lecture of different IT units in both universities to invite students to participate. The researchers collected the contact details of the students who showed interest to participate in the research study.

9.2.7 Experimental Procedure

After each of the participants signed a consent statement information form for participation in the research study, they were assigned a unique number and were provided with a small demographic questionnaire that asks about the participants' experience in software testing. Once the demographic questionnaire was completed the participants were presented with self scoring NEO PI-3 form. After the completion of the personality assessment form the participants were given the specification document of the testing task program. The testing task program was installed in a desktop computer for Swinburne University of Technology participants and in a laptop for Monash University participants. Same environment was ensured in both desktop and laptop used for the study. Participants were requested to complete the testing task within 40 minutes. However, participants took variable times. They informed when they arrived at the end of testing. They produced bug report document as a result of testing. The bug report also contained the starting and ending time. The duration was calculated from these time differences. The bug report document was later analyzed to count number of bugs found.

At the end of the test each participant received $25 for participation. The T scores and the answer form of NEO PI-3 test was kept to the researchers and the participants were provided with their personality profile form named “Your NEO Summary”.

The data was collected over the period September 2010 to October 2011.

9.2.8 Analysis

To find the association between variables we used Pearson correlation [104]. For Hypothesis testing we used a single factor multivariate analysis of variance (MANOVA) [104]. MANOVA analysis is applicable when there is more than one dependent variable. This analysis can find the effect of one or more independent variables on several dependent variables at the same time.

For testing the other null hypotheses described in Section 9.2.5 we used two sample T-Test. T-Test is used to compare the mean of two samples to examine if the means of two samples vary significantly. In conducting the first two sample T-Test we divided our main sample in to two groups depending on whether they had experience in testing or not. The first sample had experience in software testing and the second did not. For the second two sample T-Test the sample was divided in two groups depending on whether participants studied software testing related units. The first sample studied software testing related unit and the second sample did not.

We used SPSS (version 18) to conduct the MANOVA analysis and Microsoft Excel 2007 to conduct the T-Test.

9.3 Results

9.3.1 Demographic information

A total of 48 students from 18 to 35 years of age, participated in the research study. The majority of the participants (69%) were male. Almost 23% participants had professional experience in software testing. More than 30% participants had studied a specialized unit on software testing. Figure 9.3.1 shows the summary of demographic information of the participants.

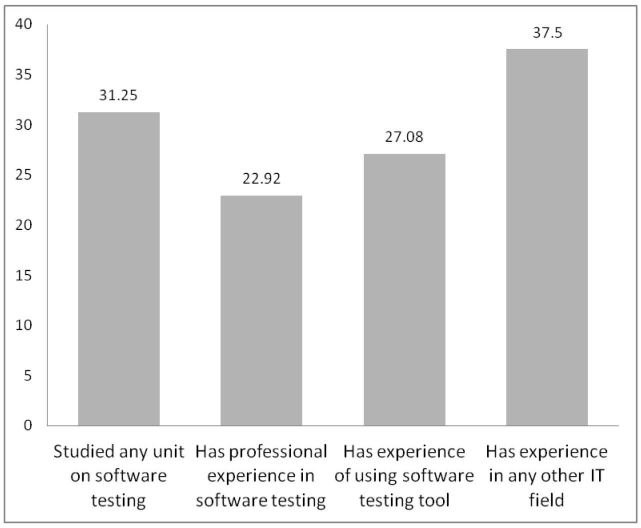


Figure 9.2: Demographic information of the participants

9.3.2 Population distribution

Table 9.3 shows the number of participants with different levels of five personality factors.

Table 9.3: Population Distribution

Factor	Number of participants					
	High (>=56)		Average (45-55)		Low (<=44)	
	Total	%	Total	%	Total	%
N	24	50	14	29.17	10	20.83
E	14	29.17	22	45.83	12	25
O	19	39.58	24	50	5	10.42
A	12	25	19	39.58	17	35.42
C	6	12.5	27	56.25	15	31.25

N= Neoroticism, E= Extraversion, O= Openness to experience, A= Agreeableness, C= Conscientiousness

The scales of NEO personality inventory tests measure factors that follow an approximately normal distribution. We have applied two well-known tests of normality, namely Kolmogorov-Smirnov Test and the Shapiro-Wilk Test to test our population distribution [104]. For a small sample size of < 50, a Shapiro-Wilk Test is more appropriate. If the Sig. value of the test is greater than the alpha level 0.05 then the data is not significantly non-normal else the data significantly deviate from a normal distribution. Table 9.4 shows the details of the test.

The Shapiro-Wilk test results presented in Table 9.4, shows our population on each of the five personality factors, does not significantly deviate from normal distribution.

Table 9.4: Tests of Normality

	Kolmogorov-Smirnov			Shapiro-Wilk		
	Statistic	df	Sig	Statistic	df	Sig
N	.134	39	.077	.963	39	.231
O	.107	39	.200	.970	39	.365
E	.100	39	.200	.965	39	.269
A	.097	39	.200	.983	39	.819
C	.118	39	.184	.969	39	.354

N= Neoroticism, E= Extraversion, O= Openness to experience, A= Agreeableness, C= Conscientiousness

9.3.3 Correlation Between Personality Traits and Effectiveness in Testing

Table 9.5 shows Pearson correlations between our dependent and independent variables. From this table we see that at significance level p = 0.05, there is a negative association between extraversion and O_{sum}, O_{wsum} and bug report quality. At the same level of significance we also see a positive, though weak association between conscientiousness and bug location rate and a weak negative association between conscientiousness and weighted fault density. There are also some correlations among different personality factors such as between Neoroticism and Conscientiousness. Since we are interested in the correlations between personality factors and effectiveness variables, we limit our discussion in this paper to those correlations only.

Table 9.5: Correlations (N = 48)

	N	E	O	A	C	O_s	O_{ws}	BLR	WFD	BRQ
N	1	-.329*	-0.136	-0.135	-.457**	0.034	0.036	-0.122	0.02	0.043
E		1	.401**	-0.231	.375**	-.267*	-.267*	0.038	-0.133	-.265*
O			1	-0.235	0.177	0.161	0.165	-0.025	-0.154	0.179
A				1	-0.121	0.167	0.173	-0.034	-0.215	0.191
C					1	0.026	0.026	.251*	-.241*	0.028
O_s						1	1.000**	.258*	0.085	.996**
O_{ws}							1	.248*	0.071	.998**
BLR								1	-.310*	0.214
WFD									1	0.028
BRQ										1

N= Neoroticism, E= Extraversion, O= Openness to experience, A= Agreeableness, C= Conscientiousness, O_s= O_{sum}, O_{ws}= O_{wsum} BLR= Bug Location Rate, WFD= Weighted Fault Density and BRQ= Bug Report Quality

*. Correlation is significant at the 0.05 level (1-tailed)

**. Correlation is significant at the 0.01 level (1-tailed)

9.3.4 Hypothesis Testing

Our single factor multivariate analysis of variance (MANOVA) results are presented in Table 9.6. We are particularly interested in the significance levels presented in the last column of Table 9.6. There are four multivariate tests conducted by default. For C_{group} effect we see Wilks' Lamda test gave $p = 0.043$, Hotelling's Trace test gave $p = 0.036$ and Roy's Largest Root test gave $p = 0.008$, all of which are less than 0.05. This means at the 95% confidence level we can reject the null hypothesis for conscientiousness and can say that based on these experimental results, there is an association between the conscientiousness trait and effectiveness in software testing. For the other four factors p value is greater than 0.05 and as such we fail to reject the null hypothesis for those personality traits.

Table 9.6: Multivariate Tests

Effect		Value	F	Hypothesis df	Error df	sig.
Intercept	Pillai's Trace	.995	414.396	3.00	6.00	0.00
	Wilks' Lamda	.005	414.396	3.00	6.00	0.00
	Hotelling's Trace	207.198	414.396	3.00	6.00	0.00
	Roy's Largest Root	207.198	414.396	3.00	6.00	0.00
N_{group}	Pillai's Trace	.793	1.534	6.000	14.000	.238
	Wilks' Lambda	.350	1.381	6.000	12.000	.298
	Hotelling's Trace	1.448	1.206	6.000	10.000	.377
	Roy's Largest Root	1.062	2.478	3.000	7.000	.146
E_{group}	Pillai's Trace	1.076	2.716	6.000	14.000	.058
	Wilks' Lambda	.188	2.607	6.000	12.000	.074
	Hotelling's Trace	2.904	2.420	6.000	10.000	.104
	Roy's Largest Root	2.293	5.349	3.000	7.000	.031
O_{group}	Pillai's Trace	.767	1.452	6.000	14.000	.264
	Wilks' Lambda	.295	1.681	6.000	12.000	.209
	Hotelling's Trace	2.176	1.813	6.000	10.000	.194
	Roy's Largest Root	2.074	4.838	3.000	7.000	.040
A_{group}	Pillai's Trace	.498	.774	6.000	14.000	.603
	Wilks' Lambda	.551	.695	6.000	12.000	.658
	Hotelling's Trace	.728	.607	6.000	10.000	.721
	Roy's Largest Root	.575	1.341	3.000	7.000	.336
C_{group}	Pillai's Trace	1.041	2.532	6.000	14.000	.071
	Wilks' Lambda	.152	3.138	6.000	12.000	.043
	Hotelling's Trace	4.329	3.607	6.000	10.000	.036
	Roy's Largest Root	4.012	9.362	3.000	7.000	.008

The results of two sample T-Test is presented in Table 9.7.

All the p-values reported in Table 9.7 are more than 0.05. That mean we fail to reject the other null hypotheses described in Section 9.2.5. As such, we did not find any significant difference in effectiveness in software testing by dividing our sample based on their experience in software testing and background knowledge of software testing.

Table 9.7: Results of T-Test

Factors	Dependent Variable	p-value
Experience in software testing	Total score (Osum)	0.28
	Weighted total score (Owsum)	0.28
	Bug location rate (BLR)	0.17
	Weighted fault density (WFD)	0.12
	Bug report quality (BRQ)	0.27
Background knowledge of software testing	Total score (Osum)	0.19
	Weighted total score (Owsum)	0.18
	Bug location rate (BLR)	0.34
	Weighted fault density (WFD)	0.43
	Bug report quality (BRQ)	0.17

9.4 Threats to Validity

In this section we discuss some of the possible threats that can attenuate the validity of the outcome of our study.

9.4.1 Threats to External validity

One possible threat to the external validity is the generalization of our findings. Since we had only limited number of participants (N=48), we cannot claim that our findings as general. We plan to replicate the study with a large number of participants in future to test the outcome of this study.

9.4.2 Threats to Internal validity

Participants were paid $25 for their participation in the study, the reward might be the motivation for participating. However, the enthusiastic participation of the participants opposes that this might have happened.

9.4.3 Threats to Construct validity

For the assessment of effectiveness in software testing, we have developed an instrument ourselves. One possible threat might be the reliability of the test. The bugs injected in the instrument represent real life bugs and match with Knuth's and Eisenstadt's classification. Hence, we can consider the instrument as a reliable instrument for testing effectiveness in software testing.

9.5 Discussion

The purpose of this study was to investigate the effect of five factor personality traits on effectiveness in software testing assessed with a custom software testing task. To find any association we closely examined the Pearson correlations of the five factors personality traits with the overall effectiveness in testing as well as different measures of effectiveness. In contrast to our assumption, we do not find much significant correlations among the personality factors and the effectiveness measures.

A weak negative correlation, however, is seen between extraversion and overall effectiveness. This differs from the results of Shoaib et al. [10] that concluded extravert people were good at exploratory testing. A negative correlation also exists between extraversion and bug location rate. As suggested by the type descriptions [88] extravert people are assertive, active and talkative. As such it is surprising that people who are active and talk much, are not good at reporting bugs. This finding should be investigated with more controlled experiment in future.

Another personality trait conscientiousness showed a weak positive correlation with bug location rate and a weak negative correlation with weighted fault density. Bug location rate takes time spend in testing in to account with the total number of bugs found in testing. According to [88] conscientious people are determined, punctual and reliable that complements the positive correlation between bug location rate and conscientiousness. On the other hand, weighted fault density is more concerned with severity of found bugs. The negative correlation of conscientiousness with weighted fault density raises the question whether highly conscientious individuals do not pay much attention to the severity of bugs. The above two findings indicate that highly conscientious individuals might be more interested towards finding more bugs in the available time without giving much attention to the severity of the bugs.

In testing the hypothesis, we found support for the influence of different levels of conscientiousness on the effectiveness in testing. Conscientiousness is closely linked to the Judging-Perceiving dimension of MBTI [131]. According to Capretz and Ahmed [13; 14], sensing and judging individuals categorized by MBTI would be good at testing. Our finding that conscientiousness has influence on testing effectiveness, thus, partially supports Capretz and Ahmed's perception. However, the exact effect of conscientiousness should be examined by more controlled experiment which is one of the plan of our future research.

Although our main focus was investigating the effect of personality on the effectiveness of software testing we also examined if there was any significant difference in experience among the participants depending on their experience and background knowledge in software testing. The results indicate that there was no significant difference in effectiveness.

9.6 Summary

In this empirical study we considered five major personality traits. Our results indicated some association of extraversion and conscientiousness with effectiveness in testing measured with different metrics. However, none of the associations were strong. We plan to replicate the experiment with larger number of participants in future to test the validity of this finding. The effect of extraversion and conscientiousness also needs to be examined with more focused experiment where the influence of the other independent variables can be kept to a minimum.

Chapter 10

Performance Appraisal of Software Testers

10.1 Introduction

The reliability of delivered software, to a large extent, depends on the performance of software testers. An accurate performance appraisal of software testers is thus very important for their recruitment, monitoring and development, and for testing team performance management. Furthermore, from our own research perspective, to conduct studies of factors that potentially affect software testers' performance, a validated, reliable instrument to assess software testers' performance is an essential prerequisite for us, and other researchers, to be able to use.

Current methods utilized to appraise software testers in the software industry have not been reported or evaluated in any detail in the open literature [138]. This gap in the existing literature - and practice - encouraged us to design the research study discussed in this chapter. This study was designed to obtain the opinions of software development project managers on the current practice of performance appraisal of software testers and the pros and cons of their current appraisal system. These helped us to answer research questions "RQ5: How is the performance of software tester appraised?" and "RQ6: How the performance of software testers should be appraised?".

Based on our review of the literature and analysis of different requirements listed in job advertisements for testers, we designed a new Performance Appraisal Form (PAF) for software testers. We also aimed to collect feedback on a proposed PAF for software testers, by requesting the software development project managers to review the form and seeking feedback on the form.

10.2 Methodology

A personal opinion survey [103] was used to conduct our survey. Compared to other available research methods, a survey enabled us to collect opinions of higher number of participants in limited available time [100]. The survey was designed by following the steps as suggested by Kitchenham and Pfleeger [103], and these are presented in the following subsections.

10.2.1 Setting the Objectives

The two main objectives of the research were to (i) collect information about the state of practice of performance appraisal of software testers; and (ii) collect feedback on a proposed Performance Appraisal Form (PAF) for software testers. Accordingly, the research was divided in two sections.

10.2.2 Survey Design

We used a web-based survey and prepared a data collection tool containing a self-administered questionnaire. The benefit of using a web-based, self-administered questionnaire is that the participants can respond at their own convenience. A potential disadvantage of web-based survey is the generalization of the sample.

10.2.3 Development of Survey Instrument

According to the suggestion of Kitchenham and Pfleeger [103], we searched for relevant research studies in the literature before developing our own survey instrument. However, we did not find any research study we could adopt some questionnaire or designing tips from. As such we developed our own instrument. Since our aim was to collect opinion and feedback, most of the data collected was in the form of open-ended free-form text.

10.2.4 Questionnaire Design

Opinion on state of practice: The section asked whether there is a formal performance appraisal process for employees, is there any specialized appraisal process for software testers or not, whether the performance appraisal of software tester in practice was considered sufficient and appropriate or not and in their opinion how software testers performance should be appraised.

Feedback on proposed PAF: In our initial design we planned to evaluate the proposed PAF using the following two steps: *Step 1:* We requested participants to consider a software tester worked under their supervision. We asked them to rate the software tester's overall performance using a scale of five with rating labels of poor, marginal, satisfactory, good and outstanding. *Step 2:* The managers were then asked to rate the same software tester's performance using our proposed appraisal form. The managers were requested to rate more than one tester working in their supervision by repeating the steps above. The overall score obtained from using our form and the overall score the manager had assigned before completing our form would be compared to check the validity of our proposed form in obtaining the right appraisal score. Once managers rate their testers, they would be given a survey form where the managers could give their feedback about our proposed appraisal form.

Due to a poor response rate to our initial PAF survey we had to revise this process and plan a more lightweight survey based on feedback from participants and potential participants that the original was too detailed and too time-consuming. We modified our survey design and participants were no longer requested to use our proposed performance appraisal form. The form was presented to the participants to review followed by a simplified feedback questionnaire asking them whether they thought the performance dimensions considered in the proposed PAF were sufficient and appropriate or not, how the proposed PAF could be improved, whether the weight assignment to different attributes were appropriate or not and so on. Most of the feedback questions used closed Likert scale responses.

10.2.5 Evaluation of Survey Instrument

To evaluate the survey instrument 11 participants including professional software engineers, academics teaching software engineering and PhD students with prior industry experience were requested to participate in a pilot survey. Based on the feedback of six participants (54.5%) who responded, definition of a severity class was modified.

10.2.6 Sampling to Obtain Valid Data

We used cluster and snowball sampling in this survey [100]. In cluster sampling, instead of selecting individuals from the population randomly, clusters of individuals are selected and within one cluster all individuals are included in the sample. In applying cluster sampling, permission to send invitation email was requested from the 12 LinkedIn and 12 Yahoo! groups that had approved us in our preliminary survey [121; 122; 138]. Three LinkedIn and four Yahoo! groups permitted us to post to the group this time, making the group response rate 25% and 33.3%, respectively. It is, however, impossible to calculate an accurate individual response rate, since most of the invitations were sent to the groups and the number of group members who actively read emails cannot be obtained.

Snowball sampling, on the other hand, is a process where samples are selected through references. The authors invited participants from their personal contacts and requested the invited participants to nominate more participants. Unfortunately we did not find any participant in the initial survey, however obtained 3 participants in lightweight survey with snowball sampling.

10.2.7 Data Analysis

We used grounded theory [100] to analyse open-ended responses. In this analysis process the researchers read the data multiple times and assign codes to the data according to the interpretation made by the researchers. Similar codes are then grouped together to form categories and categories are analysed to develop a hierarchy. The key concepts are found from the hierarchy.

Before reporting the results, a brief description of the proposed PAF is given in the following subsection. The PAF itself is available at:

http://www.testingsurveys.org/PAF_static/initialPaf.html

10.2.8 Proposed Performance Appraisal Form (PAF)

The objective of our proposed PAF is to provide a standard assessment instrument to assess overall performance of software testers from different performance dimensions. Some performance appraisal instruments use multiple forms to assess different aspects of employee performance. However, for simplicity we chose to design an integrated form. The performance dimensions of our proposed PAF were based on different approaches [92] to performance appraisal: *Performer focused appraisal* - This approach attempts to discern whether some qualities are exhibited by the performer or not. *Work behaviour based appraisal* - This approach judges the performance on the work behaviour of the performer. *Result focused appraisal:* This approach includes assessment of performance based on predefined goals and objectives.

For software testers, how effectively testing has been carried out and how efficiently the testing contributed to the reliability of the software, are important. We also believe there are some general skills that are important to be high performing software testers. The appraisal form, therefore include different rating dimension on work behaviour, work outcome and personal attributes, with seven dimensions in total.

Dimensions related to work outcome: Kaner [17] emphasized on the qualitative assessment of bug report based on- ease of understanding, sufficient information to replicate the bug, short and precise description, absence of unnecessary information and using polite tone for communication. We designed two performance dimensions on the quality of bug report according to the suggestion of Kaner.

Dimension 1- Bug report (ease of understanding): This dimension helps to assess the qualities of the bug reports that are important to make those understandable. The dimension uses evaluation concept scale labels [92] with five choices since this scale is designed to evaluate the quality of the attribute.

Dimension 2- Bug report (ease of replication): This dimension evaluates the presence of sufficient information to replicate the bugs addressed in the reports. This dimension also uses evaluation concept scale labels for the same reason as the previous dimension.

We have designed two performance dimensions related to the number of bugs found. However, taking into account the drawbacks as outlined by Kaner [64] of considering only raw bug count, we have incorporated two mitigating criteria - the severity of found bugs and the difficulty of finding the bugs - that produce extra context about the bug detection performance.

Dimension 3- Bug count (compared to the ease of finding the bug): The number of bugs found by a tester is directly dependent on the ease of testing the code. We have defined three levels of difficulty from our experience and assigned varying weights. The standard scale labels [92] is associated with this dimension. The standard is considered as the average number of bugs which is highly dependent on the project, so no range or number is specified and should be decided by the appraiser.

Dimension 4- Bug count (compared to the severity of found bugs): This dimension considers the severity of the found bugs in regard to the frequency of finding those. Four levels of severity are adopted from [133]. The weights of the different levels of severity are assigned from our experience.

Dimensions related to work behaviour: We have collected and analysed the job descriptions of software testers in the popular recruitment web site [116] over a period of five days. We found the responsibilities of testers can be classified in two broad classes - test planning and execution of tests.

Dimension 5- Assessment of performance in test planning: This dimension uses frequency scale labels [92] with five choices. Since this dimension is related to work behaviour and frequency scale helps to assess how often certain behaviour is displayed, frequency scale was considered most appropriate for this dimension.

Dimension 6 - Assessment of performance in executing tests: This dimension also uses frequency scale labels with five choices for the same reason as stated above.

Dimensions related to personal attributes: We have listed the soft-skills or qualities mentioned in job advertisements of software testers in recruitment web site[116]. We have also reviewed related literature and found skills like good domain knowledge [121], [9] are important. We have designed the seventh dimension with these personal attributes of a software tester.

Dimension 7- Personal attributes of a tester: This dimension uses compare against a standard scale labels [92] to assess whether the software tester possesses the following personal attributes: domain knowledge, adaptability to new tools and techniques for testing, communication skill, attention to details and ability to handle complex technical aspects. Different weights are assigned to those attributes from our experience.

10.3 Results

Our survey was divided into two main sections. We noticed that we obtained a different rate of participation for the two sections. We found that approximately 20% of the participants dropped out after completing the first section. Thus we describe the results obtained from each section separately.

Table 10.1: Gender and nationality of the participants

Criteria	Number of participants
Gender	
Female	3
Male	15
Nationality	
Bangladesh	5
Australia	4
Canada, United Kingdom	2 each
China, Egypt, Hungary, Romania, United States of America	1 each
Employment	
Employed in an IT organization	13
Employed in a non-IT organization	4
Self employed	1
Experience of managing testers	
Less than a year	3
Between 1 and 3 years	7
Between 3 and 5 years	4
More than 5 years	4

A total of 18 participants (8 in the initial survey and 10 in the new lightweight survey) participated in the first section. Due to the rate of participation, the results are not presented with percentages in this chapter.

10.3.1 Demographic Information

The demographic information of the participants is summarized in Table 10.1.

The majority of our participants in our sample were male. This is not surprising since the majority of practitioners in the IT field are male [123]. We assume the gender ratio is similar for software testers since female participants were in the minority in all our previous studies [121; 122; 138]. Participants were distributed over a number of countries, with the most coming from (in order of frequency) Bangladesh (5), Australia (4), Canada (2) and United Kingdom (2). There were also participants from China (1), Egypt (1), United States of America (1), Hungary (1) and Romania (1) as shown in Table 10.1. Most participants (13) worked in IT organizations with little (4) working in non-IT organization and one being self employed. Since the intended participants of this survey were software development project managers, we did not explicitly ask the role of the participants. Seven participants had between 1 to 3 years experience, four had between 3 to 5 years of experience, four had more than 5 yesrs of experience in managing testers.

10.3.2 State of Practice

10.3.2.1 Current performance appraisal approaches for software testers

We asked the participants whether the organization they work in practiced a formal process of employee appraisal and whether there is any specialized performance appraisal method/form used for software testers. The responses to these questions are summarized in Figure 10.1.

From Figure 10.1 we can see that 78% of the organizations our participants came from, conduct a formal process of employee appraisal. A similar number of organizations (77.8%) have specialised processes or methods for performance appraisal of software testers. For these we asked whether they thought the specialized process or method was adequate and sufficient. We also asked whether the specialized method or process were designed for custom use by the organization. 50% of those participants (77.8% of total) indicated that their specialized

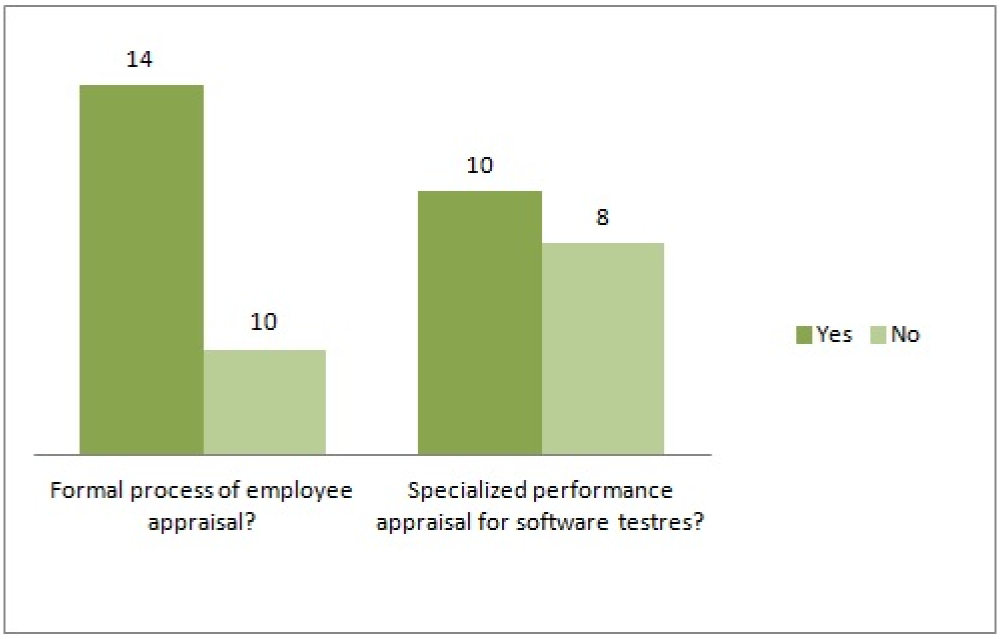


Figure 10.1: Responses on performance appraisal practice

performance appraisal process or method was designed specifically for their organization. All were satisfied with the customized method. However, responses were divided on the sufficiency of performance components considered in the specialized process or method. Only 50% mentioned the performance components were sufficient.

On the other hand, those who reported that they did not have a specialized appraisal method or process for software testers were asked how performance of software testers is appraised in those organizations. Different views were obtained in response to this question. The responses are shown in Figure 10.3.

From the responses, we see that about half of the organizations that do not have any specialized performance appraisal process or method for their software testers use a "general appraisal method". Sometimes the same appraisal method

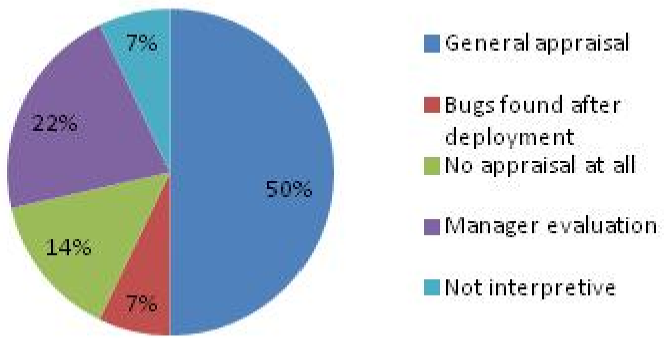


Figure 10.2: Responses on "How performance of software testers is appraised"

was used for software testers and programmers, sometimes a common HR appraisal policy was followed for all employees. These are grouped under "general appraisal method". The second most common practice was "manager evaluation". Different methods of manager evaluation were described by our participants. These include managers setting goals and evaluating the performance based on these goals; managers sitting closely by the software testers and evaluating them based on their activity; and software testers evaluating themselves and manager evaluation taking place afterwards. Surprisingly, in 14% of those organizations, no appraisal method at all is practiced for software testers. Only one participant mentioned that in their organisation software testers' performance was evaluated based on the bugs found in the live environment.

10.3.2.2 Suggestions on how software testers' performance can be appraised

In total 17 participants gave their views on how the performance of software testers can be appraised. Their responses were broad and detailed. We grouped the responses to form different categories as discussed below:

General appraisal approach: Three participants were reluctant to use specialized performance appraisal for software testers. One of them suggested using same template and same agreed upon criteria for software testers and devlopers. According to the participant this approach is "fair" and agreed upon Key Performance Indicators (KPIs) help employees to check their progress periodically. Two other participants of this group suggested designing performance appraisal for all employees based on the organization's business goals. They noted that the benefits of this approach are a sense of equal contribution and benefits and devloping required expertise for the organization's business. One participant however noted that this approach is missing "key items" for different practitioners.

Experiment based approach: One participant suggested to compare the number and severity of bugs found by different testers testing same version of software in same environment in a controlled experiment. The participant however, was not much confident about the accuracy of the process. This approach seems to be more appropriate for academic research purposes and less feasible in an industry context.

Specific performance criteria-based approach: Six participants proposed different performance criteria for appraisal of software testers, including:

using a standard performance appraisal model with software testing specific KPIs; basing appraisal on delivered system performance; using performance evaluation tools; manually appraising software testers performance; using a website where testers can fill out worklog data everyday and managers can evaluate performance based on the data submitted by the software tester; agreed upon team values and personal goals, quality of the tested product (can be accounted for by the number of bugs that reaches the client), the depth of thinking process behind discovery of a bug; time spent in regards to the type of bug (cosmetic or critical); and ability to go in depth intto the code to find bug, usage of formal QA/QC process, innovativeness of testing requirement analysis and depth of testing.

One participant suggested software testers' performance should be assessed based on their skill on testing methodology and technical knowledge, efficiency of executing tests and knowledge of the product. The participant was not specific about how software testers can be assessed based on these criteria.

Another participant suggested assessing the performance based on some common performance criteria and emphasized on the improvement of individual skill. However, he did not detail the performance criteria. They proposed that performance feedback, including achievements and limitations, should come from every party involved and should be communicated to the individual. He also suggested that the performance appraisal process should be periodical, and shorter periods are better than longer ones. However, he also noted that obtaining feedback from multiple sources adds extra work overhead and often can be difficult to be collected in time.

Manager evaluation approach: Two participants suggested the manager should work closely with the software tester and take note of every achievement and mistake they make. One of them thought since there is no process, software testers cannot deceive it. However, this approach depends on the manager, is highly labour-intensive, and is prone to bias.

Performance metrics based approach: One participant suggested the following performance metrics in order of preference: number of bugs in delivered software (lower the better); throughput - test cases run/analysed (higher the better); bug report quality; severity of found bugs (more severe, better); number of bugs found (higher better); and teamwork. The participant noted that the scheme they proposed uses specific metrics and would take time to be implemented. However, the advantage is that it focuses on the things that really matters and deemphasises quantity in favour of quality.

Another participant proposed certain metrics along with specific formulae to assess performance based on the metrics. The proposed metrics of the participant are: T: Time to run test suite; F: flakiness of tests (buggy tests); C: behavioural or functional or requirement coverage; and S: complete number of automated test cases. According to the participant the base score would be a normalized (S-x*F)*C/T, where x is a weight assigned based on the impact of flaky tests. All of the metrics proposed by the participant were related to automated testing. The rationale behind this given by the participant was that automated testing is highly desirable and that this proposed scheme will encourage more automation. The drawbacks on the other hand are that some of the metrics, such as C, are not exact. Also, optimizing a test suite for such proposed metrics can lead to designing testing that runs quickly without verifying more error prone areas of the software.

One participant proposed the following metrics: test efficiency (number of test cases executed in regards to time); number of bugs found; number of bugs found by customer; efficiency of designed test suite (test cases taken to find bugs); troubleshooting skills; and skill of communication with developers. The participant indicated although the proposed scheme can evaluate comprehensive quality of a software tester, it does not take innovativeness of test suite design into account.

From the detailed responses described above we see two major themes. One group of participants thought that software testers' performance should be appraised using the same process as is practiced for others. The other group of participants thought the opposite and advocated for a specialized performance appraisal method for software testers. Some participants of later group precisely described what criteria should be considered for performance appraisal of software testers. Different performance criteria were stated in all the reponses as indicated in Table 10.2 along with respective frequency of occurences.

From the responses, we see that about half of the organizations that do not have any specialized performance appraisal process or method for their software testers use a "general appraisal method". Sometimes the same appraisal method was used for software testers and programmers, sometimes a common HR appraisal policy was followed for all employees. These are grouped under "general appraisal method". The second most common practice was "manager evaluation". Different methods of manager evaluation were described by our participants. These include managers setting goals and evaluating the performance

Table 10.2: Performance criteria with respective frequency of occurrence

Performance criteria	Number of time mentioned
Bugs found after delivery, Number of test cases run with regards to time	4 each
Agreed upon KPIs	3
Number of bugs found, Agreed upon personal goals, In depth code inspection	2 each
Bug report quality, Severity of bugs found, Teamwork, Skill on applying testing methodology, Technical knowledge, Knowledge of the product, Use of formal QC/QA methodology, Innovativeness of testing requirement analysis, Depth of testing, Thought process behind discovery of bug, Time spent on type of bugs (cosmetic vs crucial), Number of buggy tests, Number of automated test cases, Behavioural or functional or requirement coverage, Efficiency of test case design, Communication skill	1 each

based on these goals; managers sitting closely by the software testers and evaluating them based on their activity; and software testers evaluating themselves and manager evaluation taking place afterwards. Surprisingly, in two of those organizations, no appraisal method at all is practiced for software testers. Only one participant mentioned that in their organisation software testers' performance was evaluated based on the bugs found in the live environment.

10.3.3 Feedback on Proposed PAF

All of the participants who responded to the first section of the survey did not participate in the second section. We noticed a 25% and 20% drop out for the initial survey and the new lightweight survey, respectively. As a result we obtained a total of only 14 participants who completed feedback on the proposed PAF.

Responses to the first three questions from the initial and the new lightweight survey are analysed together. However, possible responses to the other questions of this section were different, so those are presented separately.

The main feedback on our new proposed PAF was whether participants considered the proposed form is appropriate for performance appraisal of software testers. The responses to this are shown in Figure 10.3.

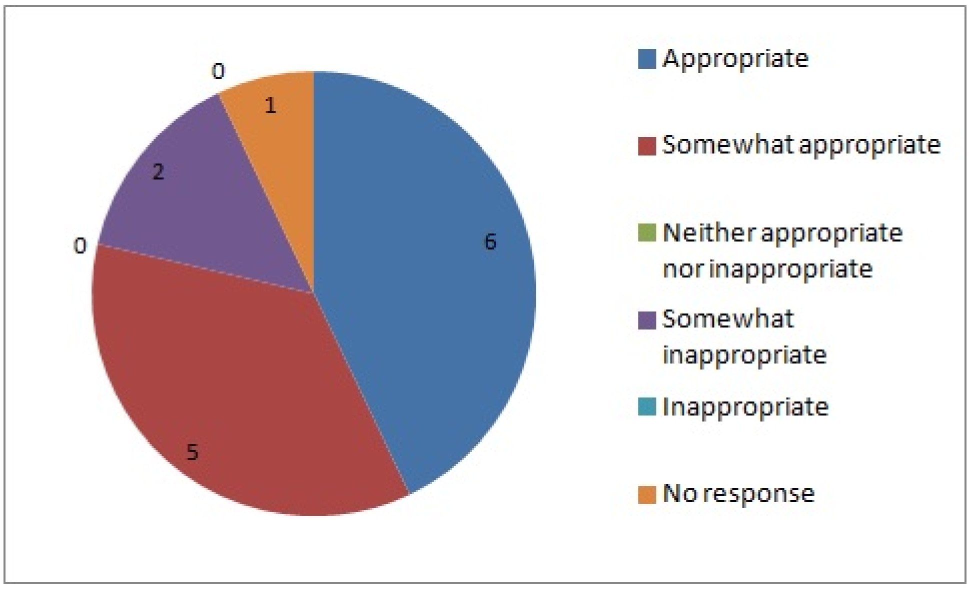


Figure 10.3: Responses on “Do you think the form is appropriate for performance appraisal of software testers?”

The majority of participants thought the proposed PAF was at least somewhat appropriate. None of the participants considered the PAF completely inappropriate. However two participants thought the PAF was somewhat inappropriate. Three participants gave their comments on the appropriateness of the PAF in the accompanying open ended question. The impression on the appropriateness of the PAF was variable. Some participants gave an overall suggestion whereas some were more specific about what to improve.

One participant thought that the form is appropriate for general performance appraisal of software testers. However, depending on the business type of the organization the form could be varied. Another participant suggested that accounting for number of bugs that pass through testing without noticing instead the number and severity of found bugs. The final participant who gave comment on this was more specific suggesting to include some evaluation about automated testing, being more specific about what a satisfactory test plan is and ignoring the number of bugs found, since software testers often report "misfeatures or UX problems" as bugs.

Our proposed PAF had a total seven performance dimensions. We asked the participants whether the dimensions we considered are sufficient or not. The responses are given in Figure 10.4. From the responses we see that, although most of the participants thought the dimensions considered were somewhat sufficient, some participants thought the opposite. We found some interesting comments from the open ended responses to this.

One participant suggested adding another dimension where testers can set their own goals in terms of capabilities and weaknesses and can self-evaluate their achievement on these. The self evaluation can help their manager's evaluation. Another participant suggested taking the innovation skill of the process into account. Another participant stated that reproducing bugs is very important and the time taken to reproduce a bug often reported by customers and requested to be fixed urgently should be considered in the PAF dimensions. Another participant suggested the ability of working in a team should be considered.

The overall score obtained by the PAF dimensions are interpreted as poor

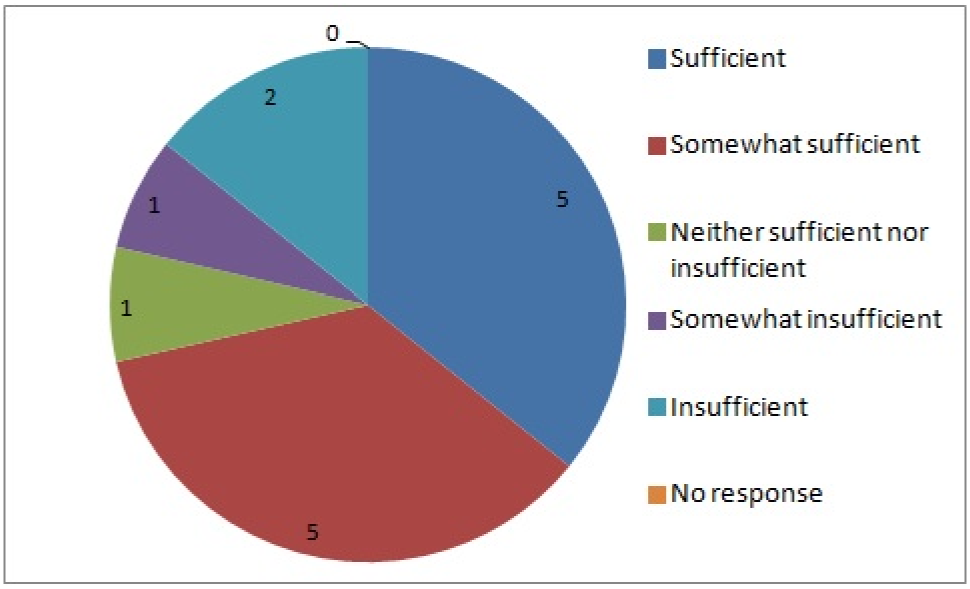


Figure 10.4: responses on "Do you think the dimensions considered in this form are sufficient to assess the performance of a software tester?"

(0-0.99), marginal (1-1.99), satisfactory (2-2.99), good (3-3.99) and outstanding (4-5). We requested participants to indicate whether they thought the proposed interpretation of the overall score is appropriate. As shown in Figure 10.5 none of the participants considered the proposed interpretation inappropriate. However, three participants were not sure about this and selected "Neither appropriate nor inappropriate". No open ended responses were received for this question.

We asked participants to indicate whether they clearly understood the performance labels attached to each performance dimension from respective definitions. In the new light weight survey they could indicate their overall response to this. However, in the initial detailed survey they could indicate the ease of understanding for each dimension.

From the responses we see that the majority of participants could understand the labels from definitions well (two participants indicated the lables were "very

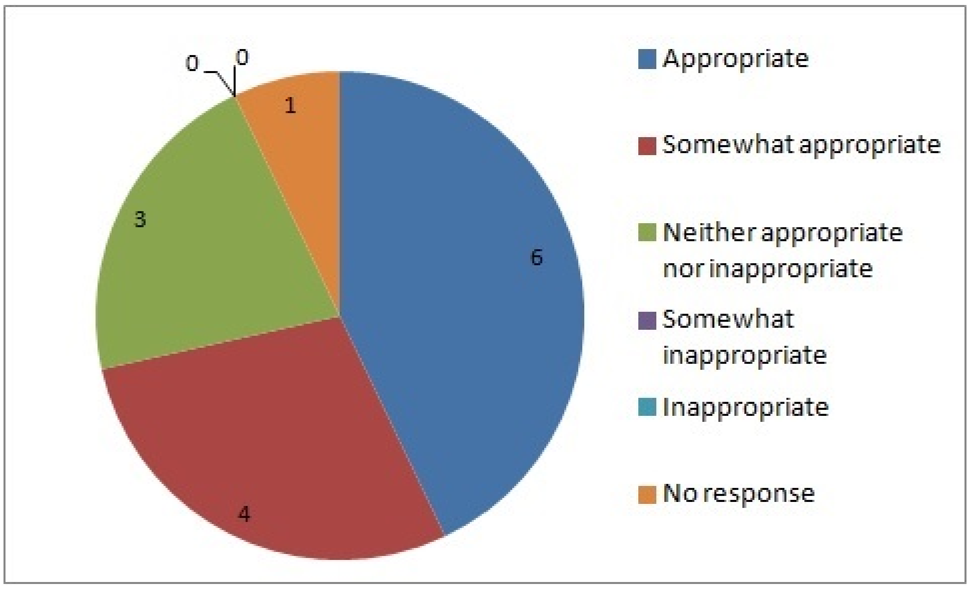


Figure 10.5: Responses on “Do you think the interpretation of overall score is appropriate?”

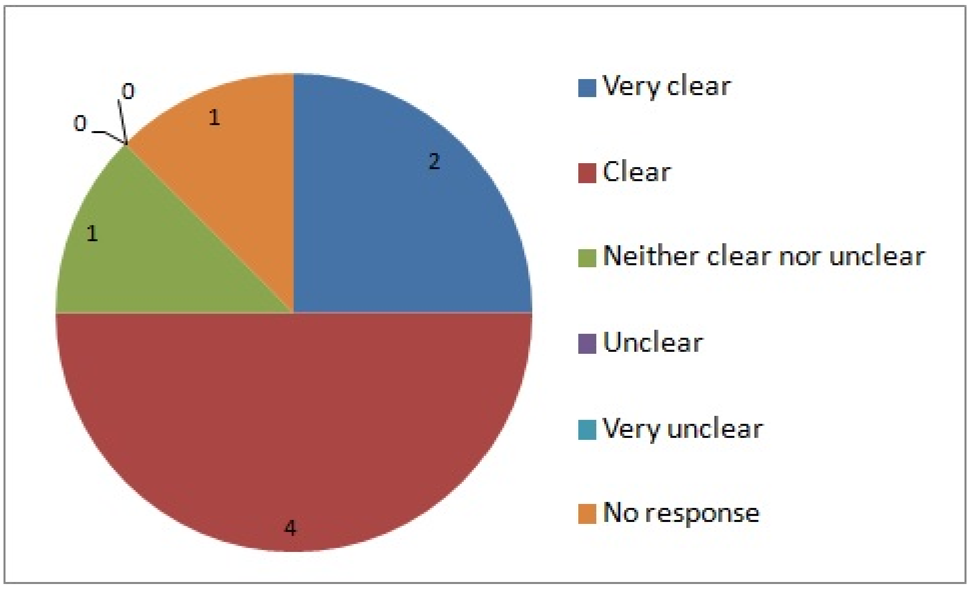


Figure 10.6: Overall understanding of the performance labels

Table 10.3: Comments on relative weight assignment

Dimension.	Description	Relative weights	Comments
3	Bug count vs ease of finding: number of bugs found in comparison to difficulty of finding	Three levels of difficulty of finding bugs	Participants views on relative weight assignments were positive. One participant stated weight assignment was good, another "alright". One stated non bug feedback are not included. Another said number of bugs found should not be compared to difficulty of finding since it depends on complexity.
4	Bug count vs severity: Number of bugs found in comparison to severity of bugs	Four levels of severity	Three participants stated relative weight assignment was "good", "useful" and "about right". Another suggested this is valuable but should not limit the number of bugs
7	Personal attributes: extent to which person being appraised possesses certain attributes	Different personal attributes	Two participants stated the relative weight assignments to this dimension was good. Two participants mentioned that this dimension is the most important.

clear", four indicated the labels were "clear"), although a few (one participant indictaed those were "neither clear nor unclear") were not sure about this. However, for the specific responses of the initial survey, we see that two participants thought the definitions were unclear for dimension 3 and 4.

Three of the performance dimensions in the proposed PAF contained relative weights assigned to different performance labels. We asked participants to comment on the relative weight assignment. The comments to each dimension with relative weights are summarized in Table 10.3. The majority of the participants were happy with the weight assignments with one participant indicating the severity of bugs found should not limit the number of bugs found.

We asked participants to indicate whether they thought that the personal attributes we considered in dimension 7 were sufficient with possible response options "yes" and "no" in the initial survey and five point Likert scale responses in light-weight survey. In the initial survey three participants thought the attributes were sufficient. The responses to this obtained in the new light weight survey are presented in Figure 10.7. Participants who thought the attributes considered were insufficient, suggested several additional attributes (in order of frequency of occurrence): ability to cooperate in a team, ability to deal with clients and colleagues, patience, ability to raise important issues to management at the right time, self organization, presentation skill, and being pedantic.

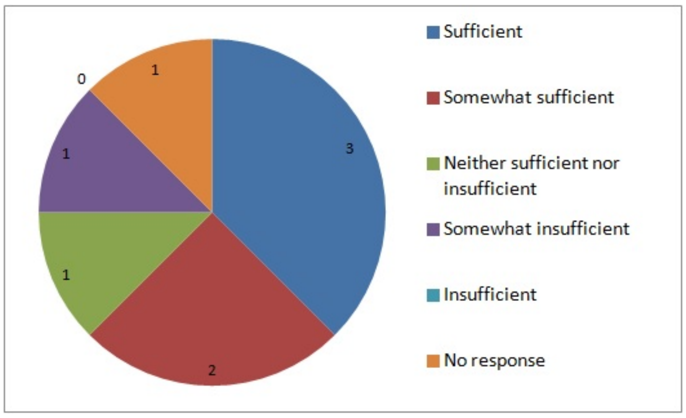


Figure 10.7: Responses on "Do you think the personal attributes considered in dimension7 are sufficient?"

10.4 Threats to Validity

One of the threats that can limit the validity of our reported results is over-generalization of the findings. We believe that the nature of participation required for this study put off many potential participants and as such we obtained a limited number of participants. Due to the small number of participants we had to modify the survey to require less time and yet still the participation rate was not satisfactory. We found that many participants were less interested in the second section and as a result we noticed a number of drop outs. Although some of our findings are interesting, due to the low number of responses we cannot strongly conclude that our finding applies in general. Another limitation of this study was modifying our proposed PAF based on the feedback we received from the participants. Compared to the first section of the survey, the responses to this section were not broad and informative enough. As a consequence of this we cannot gain much confidence on our proposed PAF. However, useful feedback was obtained on various aspects which have helped us to refine it further.

10.5 Discussion

From the responses to the survey questionnaire of the state of practice of performance appraisal, it is evident that formal process of employee appraisal is practiced in majority of organizations. A specialized appraisal process is also common for software testers. However, when there is no such specialized process, performance of software testers is most commonly appraised using a more general employee appraisal process or a manager's evaluation. In only a small number of organizations are bugs reported after deployment also considered a measure of the performance of software testers. It is the software testers responsibility to ensure certain level of reliability of the software. Bugs encountered in a live environment is an ultimate measure of the reliability of the software. We believe that this might be the reason that this criterion is used in small instances.

In response to the request to propose a performance appraisal method for software testers, participants gave detailed responses. The majority of participants advocated for specialized performance appraisal for software testers. Participants indicated different criteria for testers on which performance can be measured. Among those, counting the number of bugs encountered after delivery and the number of test cases run during unit testing were most popular. Participants also stated that KPIs should be agreed upon from the beginning and that evaluation should occur more frequently. Number of bugs found was also considered important by a few participants. As has already been discussed, this contradicts Kaner's [64] views on bug counts and our finding in the preliminary survey [138]. In the preliminary survey we also found that the severity of bugs found was a better measure than number of bugs found. However, compared to severity of the bugs found, number of bugs found were stated multiple times by our participants.

We found that our proposed PAF was considered appropriate by the majority of the participants. The performance dimensions in the proposed PAF were also considered generally sufficient. Participants suggested that we consider some extra dimensions, such as time taken to reproduce a bug and agreed upon KPIs. The responses indicate the majority of participants were satisfied with the interpretation of the overall score, the labels attached to each dimension and relative weight assignment. We refined our proposed PAF according to the overall feedback as summarised in Table 10.4. The modified version of the proposed PAF is available at:

Table 10.4: Feedback on proposed PAF and corresponding modifications

Feedback	Modification to be carried out	Comment
PAF should be designed based on business domain	No modification	The aim of the proposed PAF is to develop and refine a generalized performance appraisal form for software testers. Employers can deploy the PAF and add domain specific dimension(s) if necessary.
Consider number of bugs that pass testing without notice	No modification	Undiscovered bugs may indicate poor performance or especially hard-to-find bugs. However, testing cannot reveal all bugs. The importance of this criteria needs to be evaluated before considering this as a performance dimension.
Specific definition for satisfactory test plan	Added more text to dimension 5.	Dimension 5 - test planning: added text to give explanation. Modified text is-"DIMENSION 5- TEST PLANNING: Frequency of preparing efficient and good quality test plan. Quality and efficiency attributes of the test plan include: ability of assessing high risk area and selection of efficient test strategy. A satisfactory test plan should incorporate such test strategy that is able to test most important parts of the software in feasible time."
Ignore number of bugs	No modification.	We agree that number of bugs should not be directly used as a measure of testing performance. As such number of bugs are considered as related to severity of bugs and the difficulty of finding those.
Specific KPIs from testers themselves	Added a dimension.	A dimension is added where the manager can set some specific KPIs with the software tester at the beginning of the evaluation period. The score of this dimension and relative weight assignment (if necessary) is suggested done by the manager.
To consider innovation skill of process	No modification.	We believe the selection of appropriate and feasible test strategy covers the innovativeness of selected test process.
Time taken to reproduce a bug	No modification.	Suggested by one participant. In some organizations software testers may be requested to reproduce bugs that are reported by clients but not all, hence we didn't include directly
Unclear definition for "bug count vs ease of finding"	Simplified text	
Unclear definition for "bug count vs severity"	Simplified text	
Add more personal attributes	Added attribute.	"Team playing capability" was mentioned by multiple participants. Others, such as "ability to deal with clients and colleagues", "patience", "ability to raise important issues to management at the right time", "self organization", "presentation skill" and "being pedantic" were listed only once. If managers think some are particularly important, those can be added as specific KPIs.

http://www.testingsurveys.org/PAF_static/refinedPaf.html

10.6 Summary

This study aimed to obtain information about the state of practice of performance appraisal of software testers and to make suggestions on how the appraisal process can best be conducted. We found that there are two trends: some organizations use the same performance appraisal process for all employees, whereas some use a specialized one for software testers. Our participants suggested a number of criteria that should be considered in appraising performance of software testers. Among those number of bugs found in live environment and efficiency of running test cases were most prominent.

We also aimed to obtain feedback on our new proposed PAF for software testers and refine this based on their feedback. In spite of some dropouts in participation, we obtained good feedback on our proposed PAF. We found that parts of the proposed PAF were unclear to our participants and so we modified the text to make those parts more understandable. A few more useful performance dimensions were proposed by our participants. We added two more dimensions and one more personal attribute to address these proposals on our PAF. We believe the new refined PAF can appraise the performance of software testers appropriately. However, further industry deployment and evaluation of the PAF by managers is required to verify this. The proposed PAF will be helpful for the researchers aiming to investigate the influence of different factors on the performance of software testers as well as for industry practitioners to assess the performance of software testers for improvement, promotion and renumeration purposes.

Since the PAF is designed to assess professional software testers and the study reported in Chapter 9 was conducted with students, we could not use our proposed PAF in that study. This study was conducted later which is another reason for not using it in other research studies.

Chapter 11

Toolset

11.1 Introduction

All of our research studies have involved human participants. We conducted four internet based surveys to collect data from those participants. While some could have been undertaken with standard survey tools, some required quite specialised support for data capture. To help collection of data for these surveys we designed and developed a number of tools. This chapter describes the tools we have used for data collection.

11.2 Rationale

A number of service providers provide data collection services and software that can be used for data collection through the internet [139; 140; 141; 142]. With these, users can create and/or customize their own surveys. These services provide a platform for data collection for all, even for those with limited or no knowledge of software development. Most of these services, however, are restricted to paid services or are quite restrictive in the way data can be captured.

Surveys with straightforward questions and answers can be easily customized with most available survey development services. However, surveys requiring special data collection, such as our worklogs and performance appraisal information, are much more difficult to be setup with the existing approaches. Higher flexibility is required for these types of survey. Surveys with complex data analysis on-line, such as our personality appraisal tool, are very challenging to build with existing survey toolsets.

Another challenge of using existing services for data collection is the limited administrative control. For the surveys we conducted we required full control of the data to perform rigorous types of analysis. The necessity of setting up a survey with flexible interface and with full access to the data led us to design and develop our own survey tools.

11.3 Toolset

Since we chose to conduct internet based surveys for our data collection, all our data collection tools were designed to be used via the internet. As a result we developed four websites for collection of data for four of our surveys. All our data collection tools were hence interactive websites.

All our websites were designed using a 3-tier architecture [143]. This simple architecture was chosen since the websites were developed for collection of data for the research purpose only and did not require application of extensive business logic. Three tier architecture comprises of three main tiers called presentation tier, business logic tier and data tier. The users' browsers are the presentation tier and is considered frontend. Business logic tier and data tier are considered backend. Direct communication is performed between the frontned and backend and in the backend business logic tier communicates with the data tier. An example is shown in Figure 11.1.

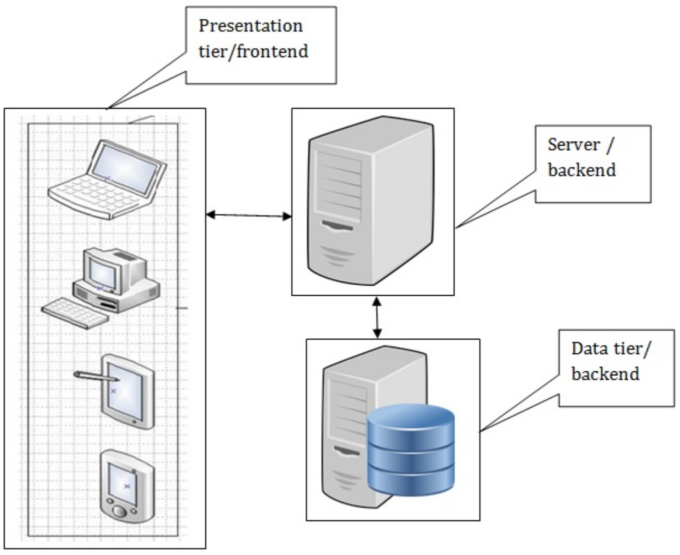


Figure 11.1: Three tier architecture

We frequently use frontend (presentation tier) and backend (business logic and data tier) to describe the data collection websites in this chapter. The frontend comprised of the user interfaces (mostly web pages) used for data collection. The backend was designed to connect to the database and to process data storage to and retrieval from the database. The detailed design and functionality of each of the websites are discussed in the following subsections.

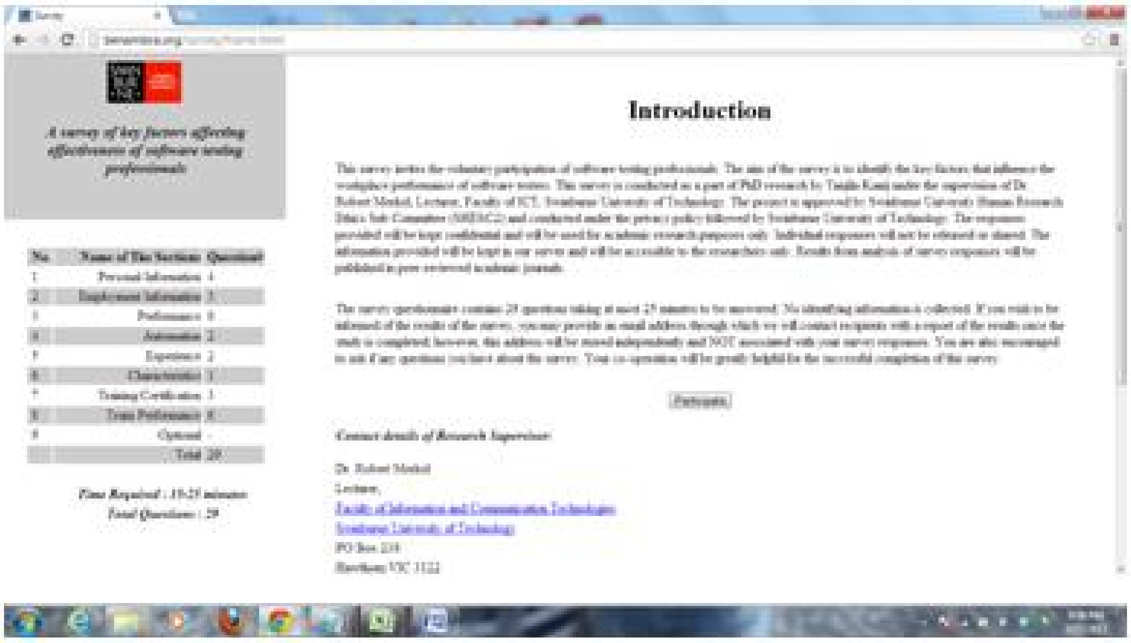
We chose to use HTML [144], PHP [145], JavaScript [146] and CSS [147] for developing the websites. We used the MySQL database [148] to store the data collected for the survey. The student researcher had prior experience of using these mark-up and scripting languages and MySQL database. As such, all the websites were developed using these familiar technologies. Another reason for choosing these languages and techniques is these are not restricted by any licensing agreements.

11.3.1 Website 1: Online Survey of Human Factors Influencing Software Testing and Testing Teams

This website was developed to collect data for our preliminary survey (discussed in Chapters 5, 6) of human factors influencing the performance of software testers. We listed a number of factors that could be influential on tester performance and asked the participants to indicate their level of agreement on the influence of these factors. We also provided some space where the participants could mention more factors (not present in our list) if they thought important. The design of the frontend and backend of this website is discussed below.

Frontend The frontend contained two sections: consent and main survey. The sections are discussed in the following subsections.

Consent section The consent information statement was signed electronically by the participants. The index page of the website contained the consent information statement with a “Participate” button. Clicking on this button indicated that the participant had agreed to the consent information statement and was referred to as signing the consent information statement electronically. Participants were taken to the main survey section through this button. A screen shot of the index webpage is shown in Figure 11.2.



A survey of key factors affecting effectiveness of software testing professionals

No.	Name of The Section	Questions
1	Personal Information	4
2	Employment Information	3
3	Performance	8
4	Automation	2
5	Experience	2
6	Characteristics	3
7	Training Certification	3
8	Team Performance	8
9	Optional	-
	Total	29

Time Required : 15-25 minutes
Total Questions : 29

Introduction

[illegible]

[illegible]

Contact details of Research Supervisor:

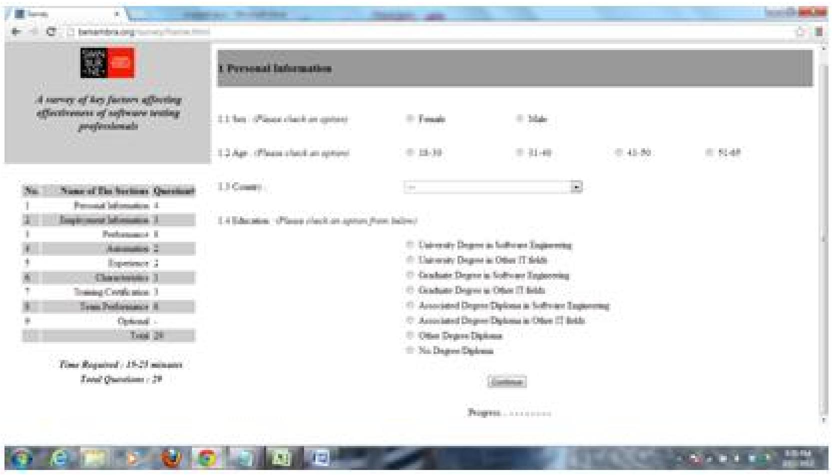
Dr. Robert Merkel
Lecturer,
Faculty of Information and Communication Technologies
Swinburne University of Technology
PO Box 218
Hawthorn VIC 3122

Figure 11.2: Screen shot of consent web page (website 1)

Main survey section The main survey was split into eight subsections: demographic, employment, performance, automated tools, experience, characteristics, trainings/certifications and team development. There were separate interface (web page) for each subsection. The web pages were presented one after another sequentially.

The main interface of the survey was divided in four HTML frames. The first frame contained the title information of the survey, the second one contained a small table with basic information (number of questions in each section) about the survey, third one loaded the web pages for different sections of the survey dynamically and the final frame called submit_frame was hidden and was used to store the submitted responses to each section temporarily before storing those to the database.

There was a progress bar on every web page to indicate what proportion of the survey questions had been completed by the participant. A screen shot of the webpage collecting demographic information of the participant is presented in Figure 11.3.



A survey of key factors affecting effectiveness of software testing professionals

No.	Name of The Sections	Questions
1	Personal Information	4
2	Employment Information	3
3	Performance	6
4	Automation	2
5	Experience	2
6	Characteristics	3
7	Training Certification	3
8	Team Performance	6
9	Optional	-
	Total	29

Time Required : 15-25 minutes
Total Questions : 29

1 Personal Information

1.1 Sex : (Please check an option) ○ Female ○ Male

1.2 Age : (Please check an option) ○ 18-30 ○ 31-40 ○ 41-50 ○ 51-65

1.3 Country : --

1.4 Education : (Please check an option from below)

○ University Degree in Software Engineering
○ University Degree in Other IT fields
○ Graduate Degree in Software Engineering
○ Graduate Degree in Other IT fields
○ Associated Degree/Diploma in Software Engineering
○ Associated Degree/Diploma in Other IT fields
○ Other Degree/Diploma
○ No Degree/Diploma

Continue

Progress . . . .

Figure 11.3: Screen shot of consent web page (website 1)

This survey accepted partial data submission. When participants filled out a section and clicked on the “Next” button, a JavaScript procedure called checkValidity() was called. This procedure checked if response was provided to all the questions. If response to any question was missing, a popup window appeared with “Carry on filling” and “Submit partial response” options. If participants chose to carry on filling, checkValidity() returned false and as a result participants were kept in the same section, otherwise checkValidity() returned a true value and another JavaScript procedure goNext() was called. This procedure submitted the responses provided on each web page to the submit_frame.

A webpage was presented at the end that provided the participants some space if they wished to write comments about the survey. Participants could enter their email addresses if they wished to get a copy of the research outcome. This webpage verified the structure of the email addresses to check if valid email addresses were provided. This webpage loaded all the data stored in the submit_frame and submitted the data along with those submitted in this webpage to the survey_submit_try.php script.

Backend: The backend of the website handled the connection and data storage to the database. These functionalities were developed in survey_submit_try.php script. A connection to the database was established and the responses to all the sections were stored to the database in this script. Email addresses were encrypted in this script before those were submitted to the database. The connection to the database was closed after the completion of the data storage to the database.

There were total 12 tables in the database. The design of the database is demonstrated with the data model shown in Figure 11.4.

The backend also contained one php script that showed a report (percentage of response to all questions along with a list of the responses to the open questions) on the collected data. This script connected with the database and retrieved the data stored in different tables to prepare the report. The script was protected by password. The administrators could enter the password in an html document. If a valid password was provided, the php script was loaded on the browser.

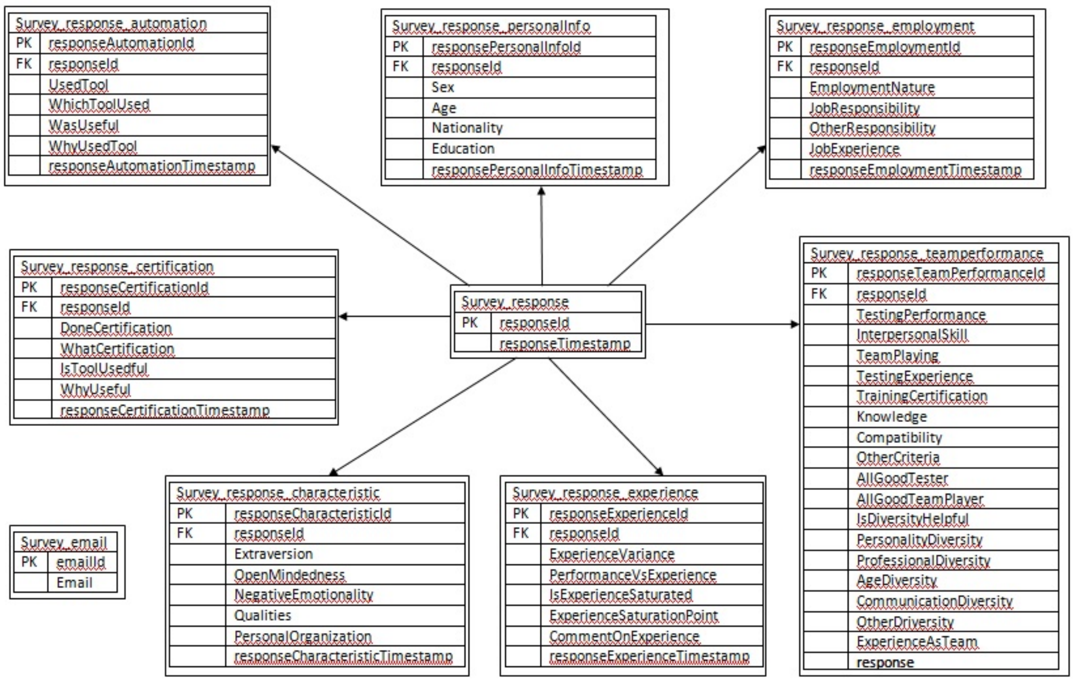


Figure 11.4: Data Model of website 1

Code repository This subsection describes the list of documents prepared for this website along with the functionalities in tabular format. There were total 17 documents in the code repository described in Table 11.1.

The website was useful for collecting the survey responses. However, after data collection was started, we noticed few database fields could not store long responses. Therefore, the maximum limit of those database fields were extended.

11.3.2 Website 2: Online Worklog Collection

This website was designed to collect worklogs from software testers. Software testers would provide their detailed time allocation to different tasks using this website. The design of the frontend and backend of the website is discussed in

Table 11.1: List of documents in web site 1

Name of the document	Functionality
frame.html	Positioned the four frames
frame1_pass_value.html	Contained title information
frame2_display_status.html	Contained other information about the survey
survey_submit.html	Initialized hidden fields to store data submitted in each section
survey_intro_try.html	Display of consent information statement and collection of consent
survey_personal_info.php	Collection of demographic information, displaying progress and submission of collected data to the survey_submit frame
Survey_employment_info.php	Collection of employment information, showing progress
survey_performance_info.php	Collection of responses to the performance subsection, showing progress and submission of collected data to the survey_submit frame
survey_automation_info.php	Collection of responses to the automated tools subsection, showing progress and submission of collected data to the survey_submit frame
survey_experience_info.php	Collection of responses to the experience subsection, showing progress and submission of collected data to the survey_submit frame
survey_characteristic_info.php	Collection of responses to the characteristics of high performing testers subsection, showing progress and submission of collected data to the survey_submit frame
survey_certification_info.php	Collection of responses to the testing related training/certification subsection, showing progress and submission of collected data to the survey_submit frame
survey_teamPerformance_info.php	Collection of responses to the testing team development subsection, showing progress and submission of collected data to the survey_submit frame
survey_optional_info.php	Collection of comments, email addresses and submission of data to backend
survey_submit_try.php	Connection to database, storing data to the database and displaying conclusion message
survey_admin_intro.html	Collection of administrative username and password
survey_admin_main.php	Verification of username and password, connection with database, retrieving data from database and preparation and displaying of report on the responses

the following subsections.

Frontend The frontend of the website contained mainly four modules- registration, login, worklog submission and message exchange module. Each module is described here.

Registration Module: Participants needed to register (create an account) before they could submit their worklog using this website. This module of the website provided the interface (web pages) for registration. The module captured basic demographic information asked as part of the survey. This module involved three main processes:

1. Collection of demographics information

2. Collection of consent

3. Account verification

Collection of demographic information: This process is implemented in the demo.php webpage. This webpage collected various demographic information about the participants. The information was collected only once while registering in the website.

A screen shot of the demographic screen is given in Figure 11.5.

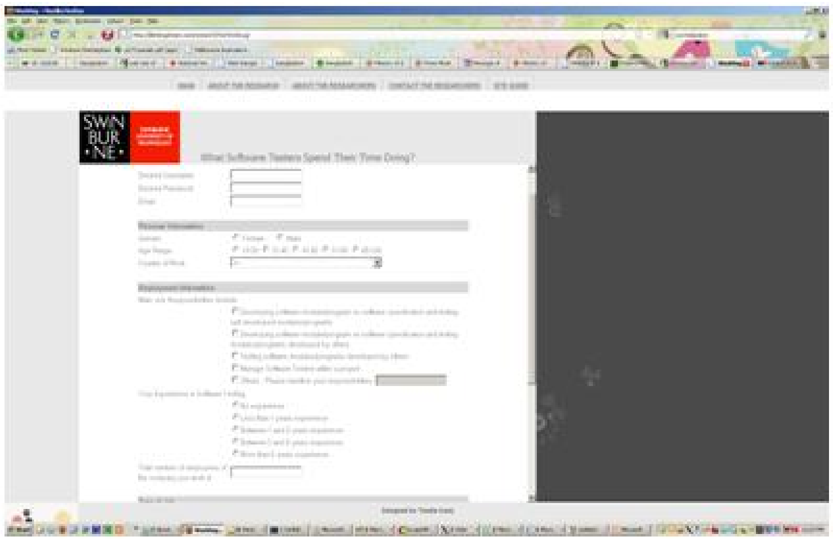


Figure 11.5: Screen shot of registration web page (web site 2)

Once the participant completed the registration form and submitted the data by clicking on the submit button, a JavaScript procedure called checkValidity() is called to verify the responses provided against some predefined constraints. If response to any of the questions was invalid the procedure focused the field where the invalid response was provided and returned false. If the procedure returned a true value then the next process (collection of consent) was triggered.

Collection of consent: Once participants clicked on the submit button in the account creation page a pop up window with the consent information statement and a checkbox with ”I give consent” appeared. Participants had to select the checkbox before they could complete the registration process. The pop up is a web page named consent.php designed with php. The consent is stored as a flag. Initially the flag is 0. Once participants checked the consent checkbox the flag was flipped to 1. The page passed the consent flag to demo.php page.

Account verification: Once all the demographic fields were filled up with valid information and the consent flag was 1, demo.php was submitted to demo_submit.php page. This page had two sections- email and data storage sections. The email section generated a unique activation key and sent email with a url (to the login module) containing this key to the email address collected in demo.php. The recipient was required to visit the url to complete the registration process. This mechanism was developed to verify the account and to avoid any fraudulent attempt at registration. The email addresses were striped and stored separately to confirm anonymity. The data storage section called database connection module(discussed later in the chapter) to connect to the database. Once a connection was established the demographic information collected through demo.php along with the activation key generated in the email section were stored in the database. There was a database field called status to keep track of the state of registration. At this stage the status field contained a value “verify” to track that the account was not verified yet. A unique database ID was generated for each registration.

Login module This module presented the main interface to login to the worklog collection website. The module contained the index webpage of the tool.

A screen shot of this page is given in Figure 11.6. This module had two main processes:

1. Login

2. Account verification

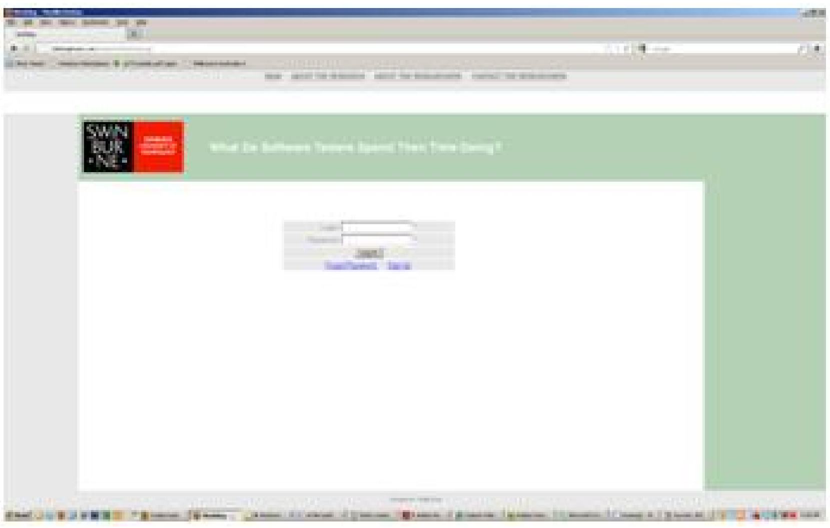


Figure 11.6: Screen shot of Login web page (web site 2)

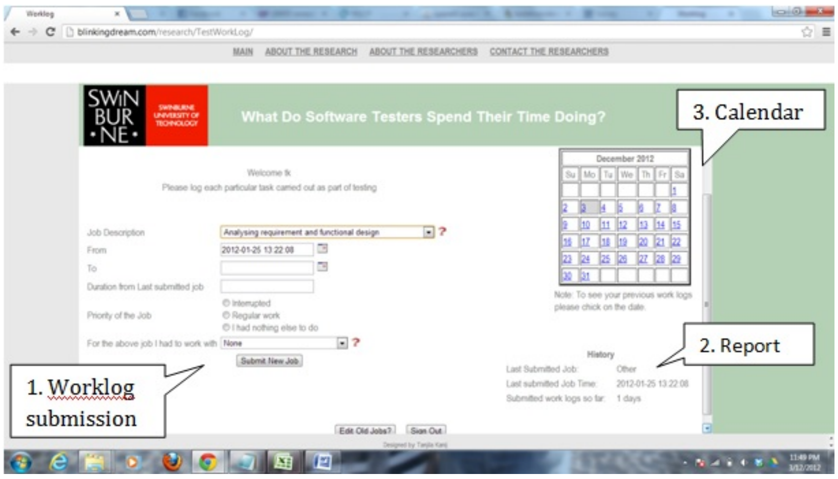
Login This process presented the participants with a web interface to provide their username and password to log in to the website. Once user entered the username and password, this process matched these with the registered value in the database. If the user was found and the status for this user was not "verify", then the participant was allowed to login to the website and was redirected to worklog submission module.

Account verification The url sent to the participants during registration redirected them to the login module. This process in the login module searched in the database for the key embedded in the url. If any record with the key was found, then the status field in that record was changed from ”verify” to “active”. In this way the account was verified and the registration process was completed. Once the status field was active, the participant could use their username and password to login to the website.

Work log Details Module Participants with an active account could access the work log module. The module was designed in a way that the participants could enter the task they had been doing in an hourly format. There were two main processes of this module:

1. Submit new job
2. Edit old job

Submit new job A drop down list contained a list of possible jobs one participant can do. This list was derived from the responsibilities of software testers mentioned in job advertisements. Participants could choose the job from the list or could write down the description of the job in their own language. Participants were asked to attach duration, priority, interactions, status and notes with the job description. The Work log main page automatically showed current date and time while entering the job description. Participants could override the time. When a job was submitted, by default, the webpage saved the duration of the work from the last submitted work to the current time unless otherwise specified. There was a field to indicate the priority of the job with choices - “interrupted”, “regular work” and “I had nothing else to do”. Participants were asked to mention if that particular job needed any kind of interaction with other team members. Only description and duration field was compulsory. This webpage contained a calendar and a report. The report showed the total hours the participants had submitted jobs for. Participants could click on any day of the calendar to see the list of submitted jobs for that particular day. A screen shot of worklog submission webpage is showed in Figure 11.7.



Welcome tk

Please log each particular task carried out as part of testing

Job Description	Analysing requirement and functional design
From	2012-01-25 13:22:08
To	
Duration from Last submitted job	
Priority of the Job	○ Interrupted ○ Regular work ○ I had nothing else to do
For the above job I had to work with	None

December 2012						
Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Note: To see your previous work logs please chick on the date.

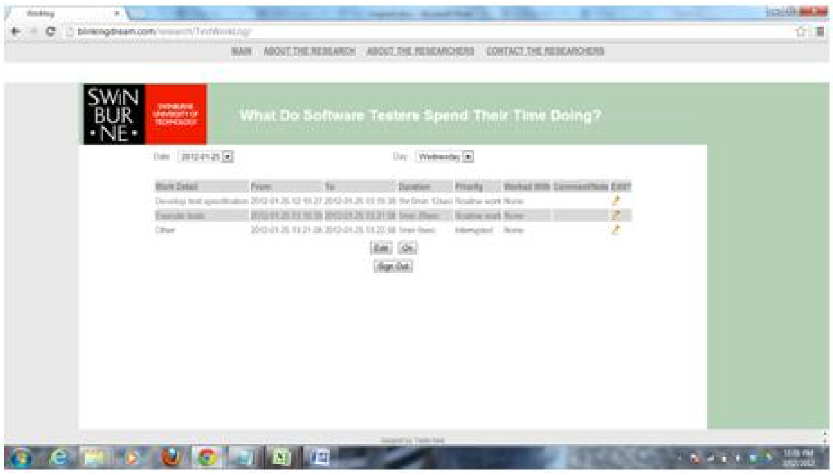
History

Last Submitted Job:	Other
Last submitted Job Time:	2012-01-25 13:22:08
Submitted work logs so far:	1 days

Figure 11.7: Screen shot of worklog submission web page (web site 2)

Edit old job A webpage containing a list of the submitted jobs by the participant was linked from the worklog submission webpage. The list could be refined by day of the week or by the dates on which jobs were submitted by the participants. If participants wished to edit any job description, they could click on the edit icon besides the job. This allowed participants to edit the description of that job. A screen shot of the list page is shown in Figure 11.8.

Message exchange Module In order to support direct communication with the researchers, the website contained a message exchange system integrated with it. Participants could send message to the researchers using the message exchange web page available from the navigations tabs. The message sent to researchers

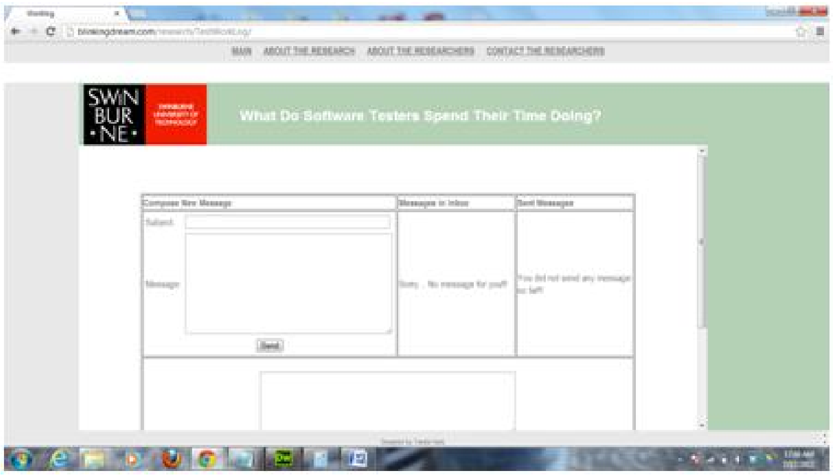


Date 2012-01-25   Day Wednesday

Work Detail	From	To	Duration	Priority	Worked With	Comment/Note	Edit?
Develop test specification	2012-01-25 12:10:27	2012-01-25 13:15:38	1hr 5min 12sec	Routine work	None		
Execute tests	2012-01-25 13:15:38	2012-01-25 13:21:58	5min 20sec	Routine work	None		
Other	2012-01-25 13:21:28	2012-01-25 13:22:58	1min 0sec	Interrupted	None		

Figure 11.8: Screen shot of edit old jobs web page (web site 2)

was stored in the database. Researchers could see the message in the database and write reply to the message in the database. The reply can be seen by the participant when they login to the website and visit the message exchange web page. A screen shot of this webpage is shown in Figure 11.9.



Sorry .. No message for you!!

You did not send any message so far!!

Figure 11.9: Screen shot of message exchange web page (web site 2)

Backend The backend mechanism for this website contained the connection to the database and storing information to the database. The database design is illustrated with data model in Figure 11.10.

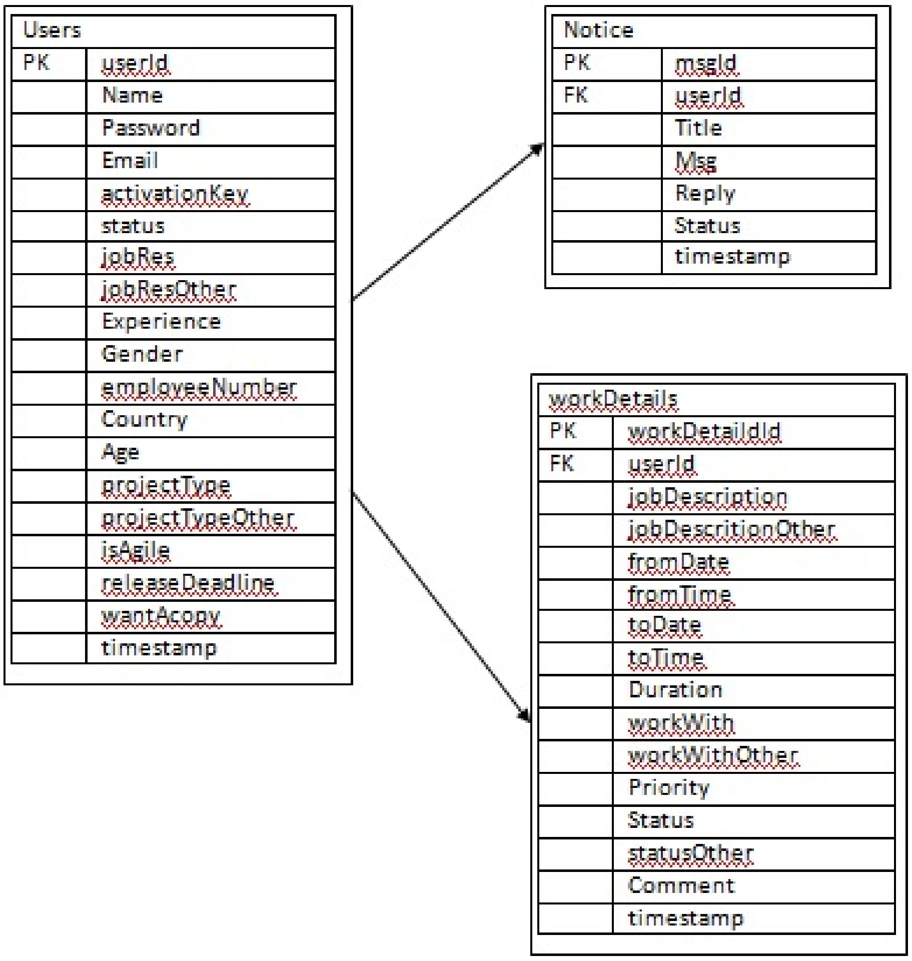


Figure 11.10: Data model of website 2

Code repository The description of code repository is given in Table 11.2.

The worklog collection website was very useful for collecting worklog. We plan to add new feature like “todo” list and time reminders to the website and design it as a small plug in that can be used by partitioners to keep track of their worklog as well as to maintain work calendar.

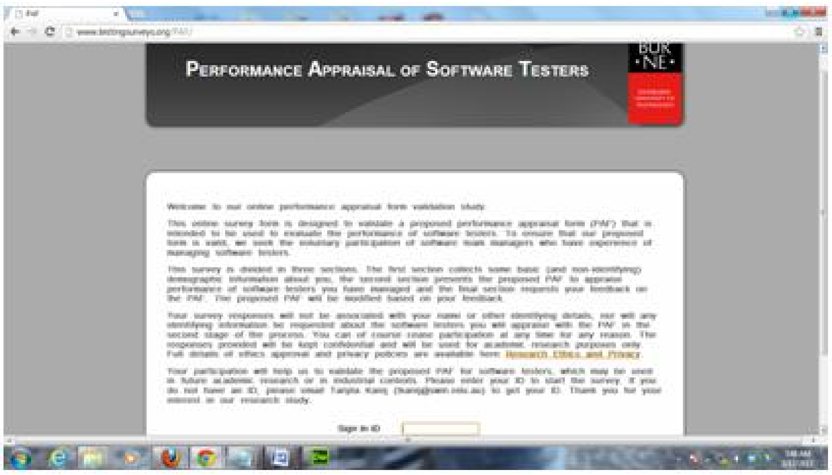
11.3.3 Website 3: Performance Appraisal Form (PAF) Validation

This website was developed to validate our proposed performance appraisal form for software testers and to help collecting data for a supporting survey of the practice of performance appraisal of software testers. More about the frontend and backend of the website is discussed in the following subsections.

Frontend The frontend of the website was divided into four sections: login, demographic and general survey, PAF experience and feedback section. The sections are discussed below.

Login section For this study the consents from the participants were collected offline. Once the consents were collected, participants were given unique ID to log in to the system. The approved IDs were already stored in the database. When participants entered their ID in the index.php web page and clicked start button, the web page was submitted to signin.php web page. This web page verified whether the ID was in the database or not. If the ID was valid (found in the database), the participant was taken to the next section, otherwise index.php was displayed with an error message. A screen shot of the index web page is shown in Figure 11.11.

Demographic and general survey section This section (demo.php web page) collected demographic information about the participants. This section also contained questions on the general practice of performance appraisal of software testers in the industry. "Next" button submitted the collected data to demo_submit.php. this web page submitted the data to the database and took



PERFORMANCE APPRAISAL OF SOFTWARE TESTERS

Welcome to our online performance appraisal form validation study.

This online survey form is designed to validate a proposed performance appraisal form (PAF) that is intended to be used to evaluate the performance of software testers. To ensure that our proposed form is valid, we seek the voluntary participation of software team managers who have experience of managing software testers.

This survey is divided in three sections. The first section collects some basic (and non-identifying) demographic information about you, the second section presents the proposed PAF to appraise performance of software testers you have managed and the final section requests your feedback on the PAF. The proposed PAF will be modified based on your feedback.

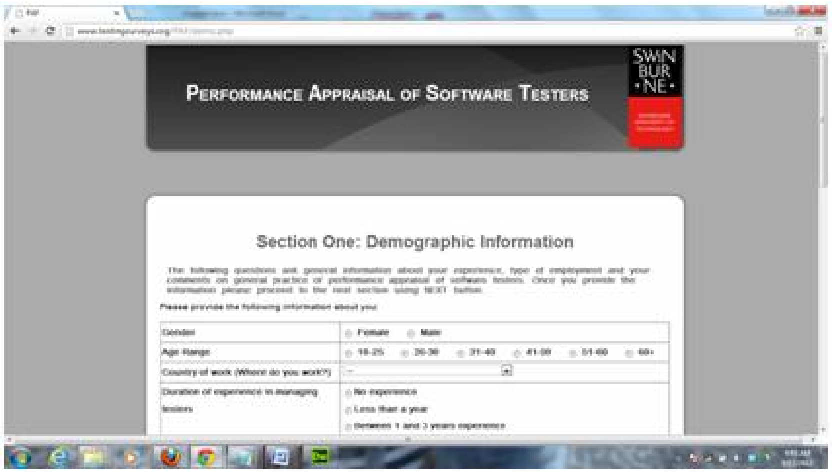
Your survey responses will not be associated with your name or other identifying details, nor will any identifying information be requested about the software testers you will appraise with the PAF in the second stage of the process. You can of course cease participation at any time for any reason. The responses provided will be kept confidential and will be used for academic research purposes only. Full details of ethics approval and privacy policies are available here: Research Ethics and Privacy.

Your participation will help us to validate the proposed PAF for software testers, which may be used in future academic research or in industrial contexts. Please enter your ID to start the survey. If you do not have an ID, please email Tanjila Kanij (tkanij@swin.edu.au) to get your ID. Thank you for your interest in our research study.

Sign in ID

Figure 11.11: Screen shot of login web page (web site 3)

participants to the next section. checkValidity() procedure in demo.php checked whether response was provided to all the questions of this section. A screen shot of this section is given in Figure 11.12.



PERFORMANCE APPRAISAL OF SOFTWARE TESTERS

Section One: Demographic Information

The following questions ask general information about your experience, type of employment and your comments on general practice of performance appraisal of software testers. Once you provide the information please proceed to the next section using NEXT button.

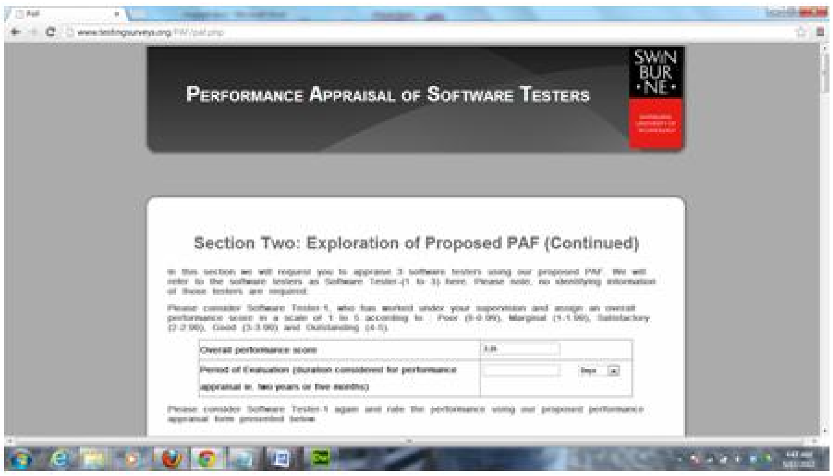
Please provide the following information about you:

Gender	○ Female ○ Male
Age Range	○ 18-25 ○ 26-30 ○ 31-40 ○ 41-50 ○ 51-60 ○ 60+
Country of work (Where do you work?)	--
Duration of experience in managing testers	○ No experience ○ Less than a year ○ Between 1 and 3 years experience

Figure 11.12: Screen shot of demographic and general survey web page (web site 3)

PAF experience section This section (paf.php web page) presented our proposed PAF to the participants to appraise the performance of at least three software testers the participants had managed. The PAF was presented three times to appraise the performance of three software testers. There was a help icon after each of the PAF items. Clicking on the icon brought a pop up window with detailed information on those items.

Participants could use the PAF to appraise more software testers if they wished to. After completion of using the PAF three times, when participants clicked on the "Next" button, a pop up window appeared and asked the participants if they wish to use the PAF more. If participants did not want to use the PAF anymore, the data were submitted to paf_submit.php and the participants were taken to the feedback section. A screen shot of PAF web page is given in Figure 11.13.


www.testingsurveys.org/PAF/paf.php
Performance Appraisal of Software Testers
SWIN BUR NE

Section Two: Exploration of Proposed PAF (Continued)

In this section we will request you to appraise 3 software testers using our proposed PAF. We will refer to the software testers as Software Tester-(1 to 3) here. Please note, no identifying information of those testers are required.

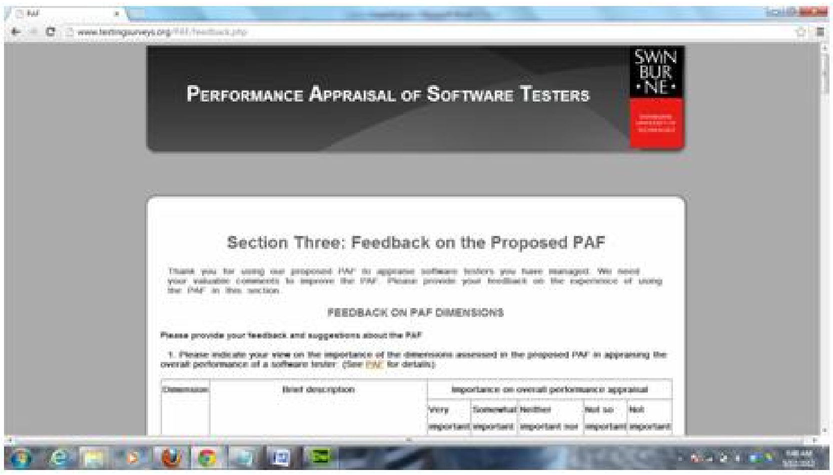
Please consider Software Tester-1, who has worked under your supervision and assign an overall performance score in a scale of 1 to 5 according to : Poor (0-0.99), Marginal (1-1.99), Satisfactory (2-2.99), Good (3-3.99) and Outstanding (4-5).

Overall performance score	3.8	
Period of Evaluation (duration considered for performance appraisal ie. two years or five months)		Days

Please consider Software Tester-1 again and rate the performance using our proposed performance appraisal form presented below.

Figure 11.13: Screen shot of PAF web page (web site 3)

Feedback section This section (feedback.php) collected the feedback on the PAF from the participants. checkValidity() procedure in feedback.php validated the responses to the questions of this section. The data were submitted to feedback_submit.php that submitted the data to the database. Participants were then taken to the last.php webpage that contained the conclusion message. A screen shot is given in Figure 11.14.


www.testingsurveys.org/PAF/feedback.php

PERFORMANCE APPRAISAL OF SOFTWARE TESTERS

Section Three: Feedback on the Proposed PAF

Thank you for using our proposed PAF to appraise software testers you have managed. We need your valuable comments to improve the PAF. Please provide your feedback on the experience of using the PAF in this section.

FEEDBACK ON PAF DIMENSIONS

Please provide your feedback and suggestions about the PAF

1. Please indicate your view on the importance of the dimensions assessed in the proposed PAF in appraising the overall performance of a software tester. (See PAF for details)

Dimension	Brief description	Importance on overall performance appraisal				
		Very important	Somewhat important	Neither important nor	Not so important	Not important

Figure 11.14: Screen shot of feedback web page (web site 3)

Backend The backend of the website comprises of the procedures for database connection and data storage to the database. A script called dbconnect.php contained the procedures for connection to the database. The database of this website contained seven tables. The data model is shown in Figure 11.15.

Code repository The documents of the website are described in Table 11.3.

The website was useful to collect the responses, however due to very small response rate, we modified the design of this survey and made it a light weighted one. In the new light weight survey, the participants no longer required any login ID. They could check the "I give consent" checkbox in the new index.php. In the new survey participants were also not requested to use our porposed PAF, instead they could review the PAF and could give feedback based on the review.

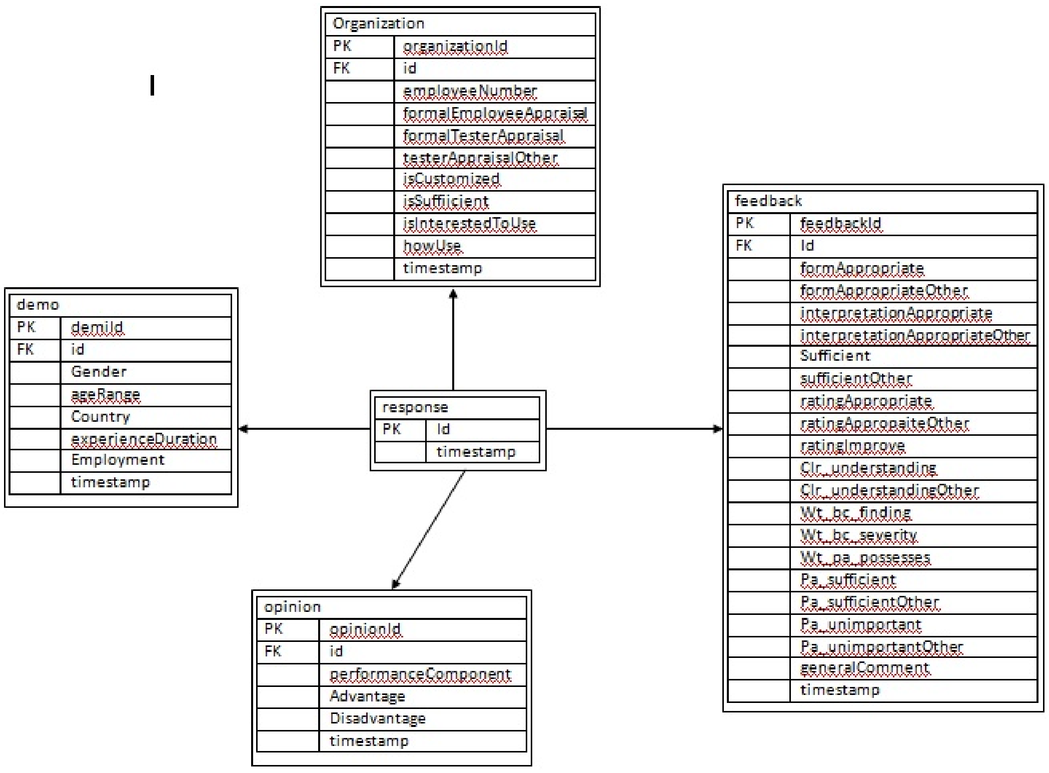


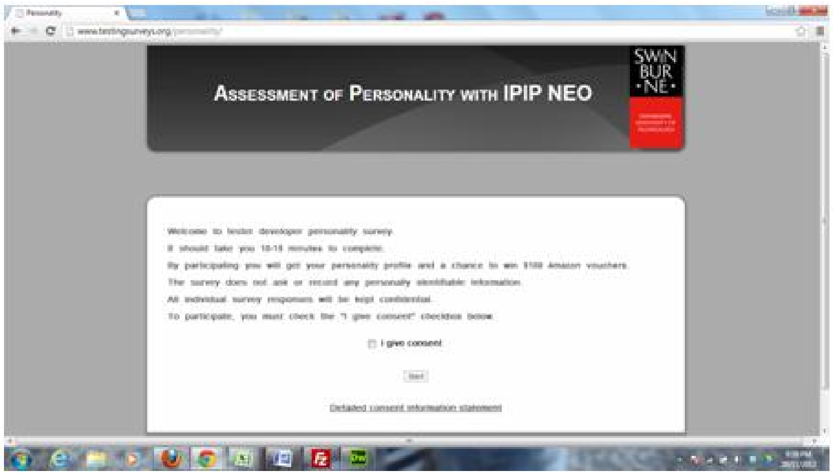
Figure 11.15: Data model of website 3

11.3.4 Website 4: Online Personality Assessment

This website was developed to assess personality of the participants. We used short version (50 items) of IPIP NEO personality assessment test for this study. There are websites available collecting responses on different version of IPIP tests. However for full administrative control over the collected responses and for using IPIP test with the smallest number of items we designed our own website. The overall design of the website is discussed below.

Frontend The frontend of the website was split in to four sections: consent, demographic, personality assessment and comment section.

Consent Section This section contains the index page of the website. This page showed brief consent information statement and a “I give consent” checkbox. Participants had to check the checkbox before they could enter the personality assessment section. “Start” button would be only active when the checkbox is checked. There is also a link to a pop up window containing detailed consent information statement. When participants clicked the start button, a unique response ID is stored in the database and a session was started. The unique response id was loaded in the session variable. A screen shot of the web page is shown in Figure 11.16.


www.testingsurveys.org/personality/
ASSESSMENT OF PERSONALITY WITH IPIP NEO
SWIN BUR NE

Welcome to tester developer personality survey.
It should take you 10-15 minutes to complete.
By participating you will get your personality profile and a chance to win $100 Amazon vouchers.
The survey does not ask or record any personally identifiable information.
All individual survey responses will be kept confidential.
To participate, you must check the "I give consent" checkbox below.

☐ I give consent

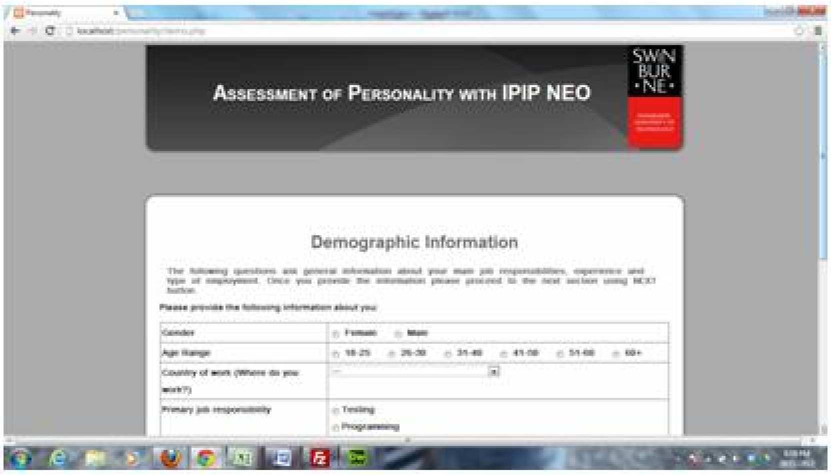
Start

Detailed consent information statement

Figure 11.16: Screen shot of consent web page (web site 4)

Demographic section This section contains demo.php webpage. Participants are taken to demo.php once they gave consent in the consent section. This section collected demographic information about the participants including-gender, age range, nationality, primary job responsibility, experience, type of employment. The “Next” button in this webpage submitted the data collected in this web page to demo_submit.php. demo_submit.php submitted the data to the database and redirected participants to the personality assessment section.

JavaScript procedure checkValidity() in demo.php, allowed the submission of data and the redirection only if all the fields on this web page were filled out with valid data. A screen shot of this web page is given in figure Figure 11.17.


ASSESSMENT OF PERSONALITY WITH IPIP NEO

Demographic Information

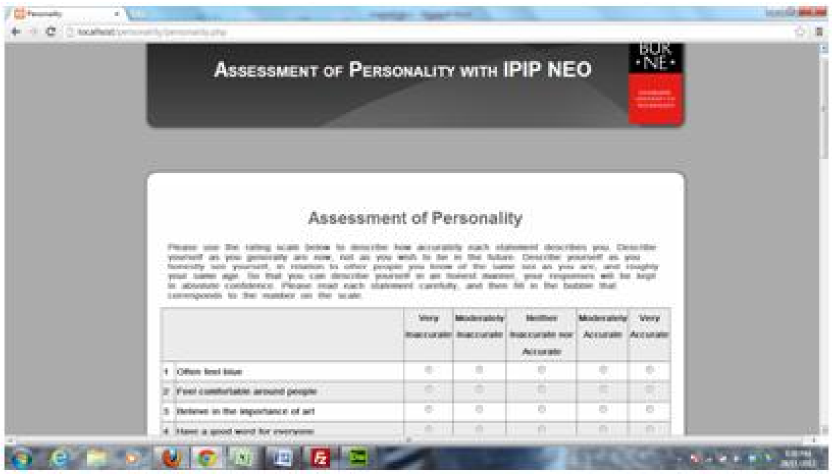
The following questions ask general information about your main job responsibilities, experience and type of employment. Once you provide the information please proceed to the next section using NEXT button.

Please provide the following information about you:

Gender	Female Male
Age Range	18-25 26-30 31-40 41-50 51-60 60+
Country of work (Where do you work?)	--
Primary job responsibility	Testing Programming

Figure 11.17: Screen shot of demographic survey web page (web site 4)

Personality assessment section This section contains the personality.php web page with the 50 items of IPIP NEO test. There were five options for response presented with radio buttons for each item. Participants had to click on the complete button after completing filling their responses to the items. checkValidity() procedure checked if all items were filled out before the responses could be submitted to personality_submit.php. personality_submit.php calculated the scores of five factors of personality from the responses to the assessment items and prepared a summary of the personality assessment outcome based on the scores. The summary was also written to a pdf document. Participants can download this pdf document if they wished to. The personality_submit.php web page submitted the responses given in personality.php to the database. A screen shot of personality.php is given in Figure 11.18.



Assessment of Personality

Please use the rating scale below to describe how accurately each statement describes you. Describe yourself as you generally are now, not as you wish to be in the future. Describe yourself as you honestly see yourself, in relation to other people you know of the same sex as you are, and roughly your same age. So that you can describe yourself in an honest manner, your responses will be kept in absolute confidence. Please read each statement carefully, and then fill in the bubble that corresponds to the number on the scale.

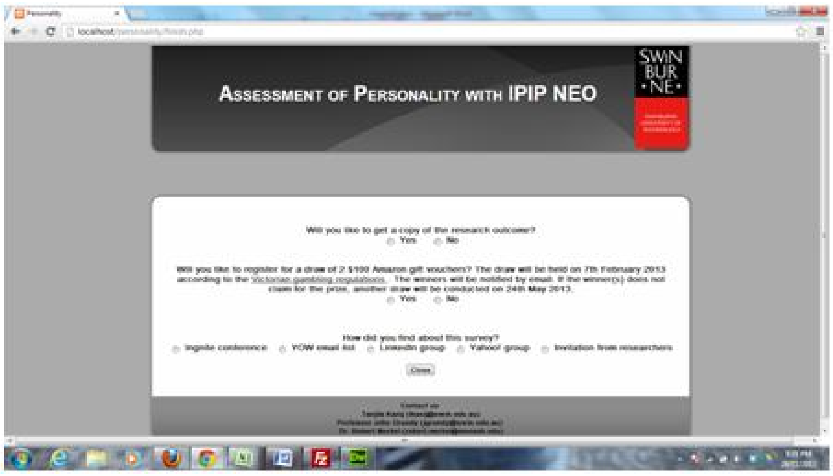
		Very Inaccurate	Moderately Inaccurate	Neither Inaccurate nor Accurate	Moderately Accurate	Very Accurate
1	Often feel blue	○	○	○	○	○
2	Feel comfortable around people	○	○	○	○	○
3	Believe in the importance of art	○	○	○	○	○
4	Have a good word for everyone	○	○	○	○	○

Figure 11.18: Screen shot of personality assessment items web page (web site 4)

Comment Section This section asks the participants whether they want a copy of the outcome of the survey and whether they want to register for a draw of prize. If participants chose "Yes" for any of the above, they were required to enter their email address. The last question of this section asked participants about the source from where they heard about the survey. By submitting these information participants entered last.php webpage that contained the thanks message for participating along with link to other research studies. A screen shot of the last screen is given in Figure 11.19.

Backend The backend of this website handled the procedures for database connection and for storing information to the database. There was a php script named dbconnect.php that contained the necessary procedures for connection to the database. The scripts that were designed to submit data to the database included dbconnect.php before submitting data to the database.

There were eight tables in the database. The database design is shown in Figure 11.20.



Will you like to get a copy of the research outcome?
Yes No

Will you like to register for a draw of 2 $100 Amazon gift vouchers? The draw will be held on 7th February 2013 according to the Victorian gambling regulations. The winners will be notified by email. If the winner(s) does not claim for the prize, another draw will be conducted on 24th May 2013.
Yes No

How did you find about this survey?
Ingnite conference YOW email list LinkedIn group Yahoo! group Invitation from researchers

Close

Contact us
[illegible]

Figure 11.19: Screen shot of Comment web page (web site 4)

Code repository The list of the documents written for this website is presented in table 11.4.

The website works as an automated personality assessment test designed based on IPIP inventory. Users of this website get their personality profile by completing the personality assessment questionnaire. Although the website was initially designed for a research study of this thesis, the website can be used for personality assessment for other purposes as well.

11.4 Summary

Existing survey development tools were found not to be able to support the range of data presentation, collection and analysis online that we required. The data collection tools described in this chapter were developed to collect data for the range of research studies designed as part of this PhD thesis. After successful completion of this thesis these tools will be published as open source in the internet to help future researchers collect data for their research studies.

Table 11.2: List of documents in web site 2

Name of the document	Functionality
index.php	Positioning frames
top_nav.html	Navigation tab
worklog.php	Collection of worklog
newUser.php	Collection of demographic information
mkcalendar.php	Preparation of calendar
about_research.html	More information about the research
about_researcher.html	More information about the researchers
submitData.php	Submission of worklog to the database
help.html	Providing more information about worklog
workDetail.php	Displaying list of submitted work log
checkLogin.php	Checking login status
common.php	Containing common php procedures
datetimepicker_css.js	Help selection duration for worklog
contact.php	Web page for message exchange
contactSubmit.php	Submission of message to the database
dbconnect.php	Connection to database
downloadconsent.php	Providing pdf document with consent information statement
forgot_pass.php	Sending new password in case of forgotten password
login.php	Initializing session variables after logging in
logout.php	Closing session variables after logging out

Table 11.3: List of documents in web site 3

Name of the document	Functionality
index.php	Displaying of brief summary about the research study and providing space to enter unique ID
signin.php	Verification of unique ID entered in index.php and redirection to appropriate web page
dbconnect.php	Establishment of connection to the database
demo.php	Collection of demographic information
demo_submit.php	Submission of demographic information to the database
Paf_index.php	Displaying general information about using the PAF
paf.php	Collection of responses to the PAF items
paf_submit.php	Submission of responses to the PAF items to the database
Feedback.php	Collection of responses to the feedback on PAF
Feedback_submit.php	Submission of responses to the feedback on PAF
finish.php	Collection of extra information (email addresses, whether wanted a copy etc.)
finish_submit.php	Submission of extra information to the database
last.php	Displaying conclusion message
Paf_show.html	Detailed information about PAF items
ethicsNpolicy.html	Displaying detailed policy of ethics approval
style.css	Style sheet for the website

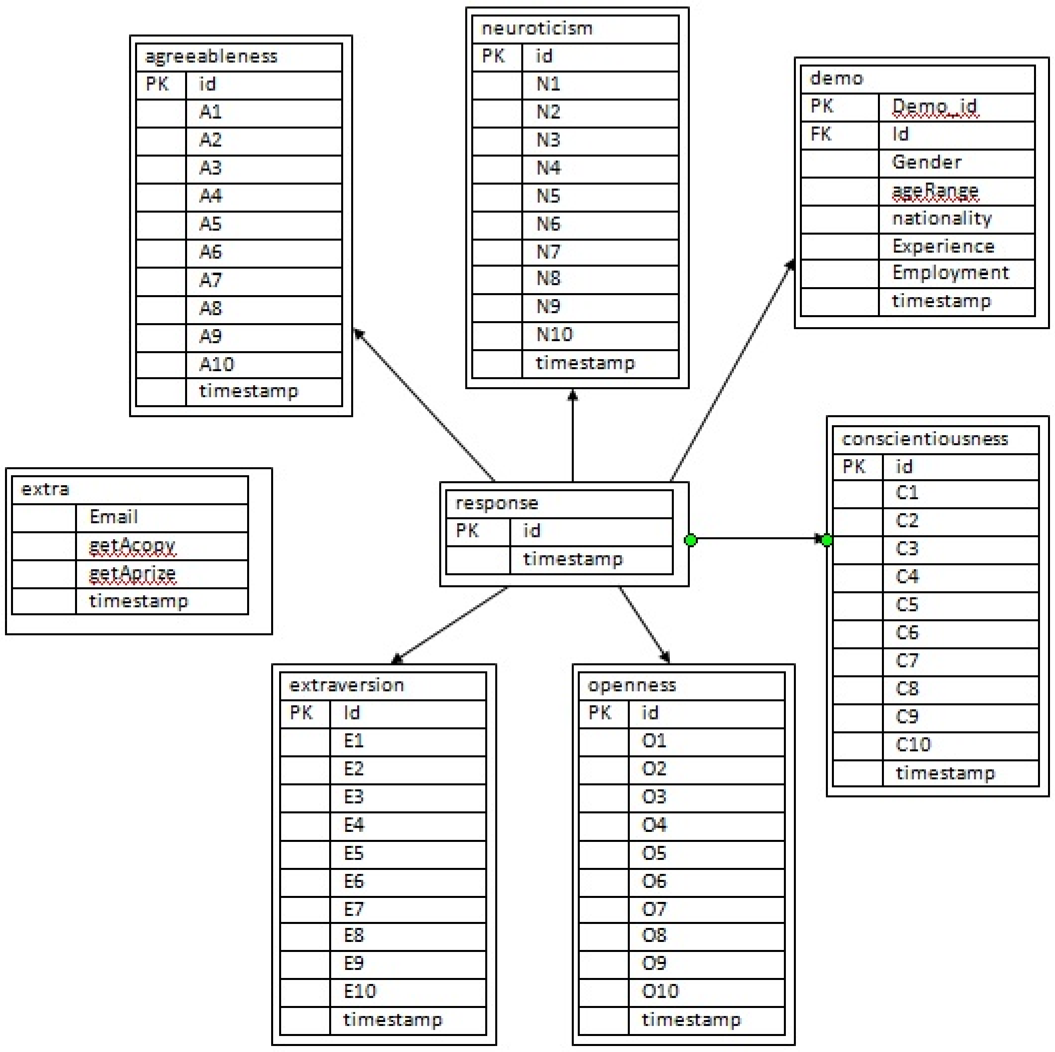


Figure 11.20: Data model of website 4

Table 11.4: List of documents in web site 4

Name of the document	Functionality
index.php	Displaying of short consent information statement and collection of consent
cis.html	Displaying detailed consent information statement
dbconnect.php	Establishment of connection to the database
demo.php	Collection of demographic information
demo_submit.php	Submission of demographic information to the database
personality.php	Collection of responses to the personality assessment items
personality_submit.php	Submission of responses to the personality assessment items to the database
finish.php	Collection of extra information (email addresses, whether wanted a copy etc.)
finish_submit.php	Submission of extra information to the database
last.php	Displaying conclusion message
fpdf.php	Contained required class for creating pdf document
fpdf.css	Style sheet for pdf document
signin.php	Collection of email address
signin_submit.php	Submission of email address to the database
style.css	Style sheet for the website

Chapter 12

Discussion

12.1 Introduction

In this chapter, we discuss our overall findings from the series of research studies conducted as part of this thesis and reported from Chapters 5 to 10. We weigh up our findings in the context of the research questions presented in Chapter 4. We also analyse possible threats that might invalidate the findings of this research. We report our experiences of conducting the empirical studies and the lessons that we have learned from our experience.

12.2 Analysis of key findings

As part of this thesis, five empirical research studies investigating different aspects of software testing were conducted from 2009 to 2012. Among the five research studies, four were web based surveys and one was a quasi experiment. The research studies were conducted with ICT students and software development practitioners. The requested participation ranged from few minutes to 15 days. The research studies are summarized below:

Study Name: Preliminary survey of factors affecting software testers performance and testing team building

Study Methodology: Survey

Participants: 104 Software development practitioners

Participants sourced from: Yahoo! and LinkedIn groups

Time required by participants: 30 minutes

Summary of findings:

1. The effectiveness of individual software testers varies considerably.
2. "Number of bugs" is not much important in measuring performance of software testers.
3. "Quality of bug report" is more important than "severity of bugs" in measuring performance of software testers.
4. "open mindedness" is a common characteristics of good software testers.
5. Usage of automation tools is very common and these are mostly used for regression testing.
6. "Testing performance", "Experience in testing" and "Knowledge of problem domain" are the most important factors for developing a testing team.

Study Name: Worklog of software testers

Study Methodology: Case study and survey

Participants: 6 Software testers and managers working in industry

Participants sourced from: Invitation from personal contacts

Time required by participants: 15 days

Summary of findings:

1. Software testers perform different categories of jobs. 12 distinct categories of jobs from worklogs, 11 categories from job advertisements and 4 categories from bug repositories were found.

2. Testing related responsibilities are broken down to a number of unit tasks.

3. Testing related responsibilities are followed by reporting related responsibilities such as reporting and explaining test results.

4. Many research and development type of tasks such as requirement analysis, gaining domain knowledge, searching for tools are performed by software testers.

5. Other responsibilities of software testers include managerial, administrative and supervision related tasks.

6. "Executing tests", "Requirement analysis", "preparing test plans" and "developing test suite" were most frequent tasks.

Study Name: Personality traits of software testers

Study Methodology: Survey

Participants: 182 Software development practitioners

Participants sourced from:

1. Yahoo!, YOW! and LinkedIn groups
2. Attending conference
3. Tweets

Time required by participants: 30 minutes

Summary of findings: Software testers are more conscientious compared to other IT practitioners.

Study Name: Effect of personality on effectiveness of software testing

Study Methodology: Quasi experiment

Participants: 48 ICT students

Participants sourced from: Inviting from lectures

Time required by participants: 2 hours

Summary of findings:

1. Weak negative correlation (0.267) between extraversion and overall effectiveness in software testing.
2. Weak positive correlation (0.251) between conscientiousness and bug location rate.
3. Weak negative correlation (0.241) between conscientiousness and weighted fault density.

4. Support for the influence of conscientiousness on the effectiveness in software testing (hypothesis testing) .

Study Name: Survey of state of practice of performance appraisal of software testers and collection of feedback on PAF

Study Methodology: Survey

Participants: 18 Software development project managers

Participants sourced from:

1. Yahoo! and LinkedIn groups
2. Invitation from personal contacts

Time required by participants: 20 minutes

Summary of findings:

1. According to the majority of participants specialized performance appraisal should be used for software testers.
2. Bugs found after delivery and test cases run in unit time should be considered for performance appraisal.
3. Our proposed PAF is appropriate for performance appraisal of software tester, one new dimension, one new personal attribute and few more text were added to the PAF based on the feedback.

From our preliminary survey we found that the performance of software testing varies depending on the individual who carries out the testing. This indicates that performance to some extent is dependent on the person. This finding makes the ground strong for investigation of personal attributes that can influence the performance. Personality is prominent among those personal attributes. A detailed assessment of personality was not feasible in our preliminary survey. However, we asked participants to indicate the characteristics of high performing software testers. Participants believed that "open mindedness" was common among high performing testers. Open mindedness is related to creativity and thoroughness, which were found to be the personal characteristics exhibited by high performing software testers by Iivonen et al. [9].

Iivonen et al. [9] also found that high performing software testers tend to be conscientious, patient, persistent and accurate. Most of these characteristics are closely related to the "conscientiousness" factor of five factor model of personality. In our survey of personality traits of software testers, we found that testers were higher on conscientiousness compared to other IT practitioners. This observation complements the personal characteristics listed by Iivonen et al.

Capretz et al. [13] suggested that sensing and judging type people as assessed with MBTI will be good as software testers. These dimensions of MBTI are related to the conscientiousness factor of the five factors model of personality. Thus this result also complements our findings. In our quasi experiment examining the effect of personality traits on the effectiveness of software testers, we found that conscientiousness was positively associated with bug location rate and negatively associated with weighted fault density. Both the associations were however, weak. Bug location rate is more concerned with timely execution of tests and maximizing detection of bug in specific time frame. Since highly conscientious people accomplish tasks in systematic ways and maintain good organization, it is not a surprise that conscientiousness has a positive association with bug location rate. Weighted fault density, however, is dependent on severity of bugs. Weighted fault density might be associated with some intellectual attributes that might not be much related to conscientiousness.

Another factor, extraversion, showed a weak negative association with overall effectiveness of software testing in our quasi experiment. This is different to the finding of Shoaib et al. [10], who found extravert people were good exploratory testers. A negative correlation also exists between extraversion and bug location rate. As suggested by the type descriptions, extravert people are assertive, active and talkative. As such it is rather surprising that people who are enthusiastic about communication may not be as good as others at reporting bugs.

From this research study we found that software testers perform a variety of tasks as part of their role. These range from requirements engineering and functional design analysis to administrative tasks. We found testing specific tasks, such as writing test cases and executing tests, are often followed by reporting and explanation of test results. Research related responsibilities, such as gaining domain knowledge, selecting best testing tools and methodologies, are also performed by software testers. Software testers acquire experience in performing these tasks. This supports the view that experienced testers possess more domain knowledge and that this helps them to "fill in the gap" and to interpret unclear and ambiguous specifications, as suggested by Beer and Ramler [11].

In examining the influence of personality traits on effectiveness of software testing, we required a means to distinguish between high and average performing testers. However, such a means (an instrument or a method) was unavailable.

This gap in the available body of research encouraged us to design a performance appraisal form for software testers. We considered the proposed assessment methods by Kaner [17; 64] in designing the form.

In a survey we collected information on the state of practice of performance appraisal of software testers and feedback on our proposed performance appraisal form. From the responses we found that software testers are sometimes appraised with common employee appraisal used for other employees. However, a specialized appraisal method or instrument is also used for appraising software testers. As an outcome of the research study, we refined our proposed performance appraisal form based on the feedback received that can be used to appraise the performance of software testers.

Our initial research plan was to find specific personality traits that influence software testers' performance. As such we designed our preliminary survey to find whether the list of factors that are influential to software testing includes personality. Our second research study was investigation of any connection between personality traits and performance of software testers. However, in reviewing the relevant literature and during conducting the initially planned research studies, we came across "gaps" in existing research such as- no agreed upon list of software testing and no standard performance appraisal method. This encouraged us to design research studies to address these issues. The studies reported in Chapter 7 and Chapter 10 were conducted later, therefore the outcome of these studies could not be directly used in other studies.

The summarized finding of the series of research studies conducted are reported in this section. Some of the studies suffered from lack of enough participation. There were also some threats (discussed separately) associated with the research studies. Considering these, although information, we cannot claim our findings apply in general.

12.3 Findings in relation to the research questions

The series of research studies were designed based on the research questions listed in Chapter 4. In this section we analyse the implication of our findings of the different research studies in answering the pre defined research questions.

RQ1: What factors influence the effectiveness of software testers?

The preliminary survey reported in Chapters 5 and 6 was designed to address this research question. Before going to the detail of what factors influence the effectiveness of software testers, we wanted to determine whether or not there is a significant difference in effectiveness. The responses we obtained indicate that participants believed the effectiveness in software testing significantly differed from individual to individual. Factors such as personal characteristics, experience contributes to this difference.

RQ2: What do software testers do?

The research study reported in Chapter 7 was designed to find answers to this research question. We collected worklogs of software testers working in industry, software testing related job advertisements, and bug reports from open source bug repositories to prepare a list of tasks performed as part of software testing.

We found that software testers typically perform a wide variety of tasks. Software testing related responsibilities comprise of number of unit tasks such as test planning, developing test suite, executing tests and so on. Reporting of found bugs is often followed by explaining bugs. A number of research and development type of tasks are also performed by software testers. These include tasks such as analysis of requirement and design documents, searching for new tools and so on.

RQ3: What personality traits are over-represented among testers?

We collected personality profiles of a number of software development practitioners in the research study reported in Chapter 8. These personality profiles were prepared using IPIP NEO test designed based on the five factor model of personality. We found that software testers are significantly more conscientious than non-tester IT professionals.

RQ3.1: Are these traits different from other practitioners?

In the research study described in Chapter 8, we also collected the personality profiles of other software development practitioners. Their profiles were compared to those of software testers. The score on five major personality traits of the software testers and the non testers were compared and we found that software testers were significantly higher on conscientiousness. On other factors, testers and non-testers do not have a significant difference in personality traits.

RQ4: Do personality traits influence software testing performance?

In our experiment reported in Chapter 9, we examined the influence of personality traits on the effectiveness of software testing. We found a weak association between two of the personality traits and some of the measures of effectiveness in software testing. More specifically, we found that extraversion and overall effectiveness was negatively correlated (r=-0.267), conscientiousness was positively correlated with bug location rate and negatively correlated with weighted fault density. All these associations were, however, quite weak.

RQ4.1: If yes, which trait(s) has the maximum influence?

We hypothesized that the effectiveness of the student testers for a given software testing task would differ significantly based on the different level of five major personality traits. In the hypothesis testing we found the support for our hypothesis only for conscientiousness.

RQ5: How is the performance of software testers appraised?

We conducted a survey to collect information on the state of practice of performance appraisal of software testers. The survey is reported in Chapter 10. We found that specialized performance appraisals are often used for software testers. However, software testers' performance is also often appraised with the same appraisal method or process as is used for other IT professionals. On very rare occasions bugs reported after deployment is also considered a criterion of performance appraisal of software testers.

RQ6: How should the performance of software testers be appraised?

In the accompanying questionnaire of the survey reported in Chapter 10, we requested opinions from software development project managers on how performance of software testers' should be appraised. The participants were very enthusiastic and gave detailed opinions on this. The participants suggested different criteria to be considered for performance appraisal of software testers, including bugs found after delivery and efficiency of executing test cases. We developed a new potential performance appraisal form and process for use in appraising testers.

12.4 Threats to Validity

In this section we discuss some of possible threats that can impact the validity of our overall findings.

12.4.1 External Validity

The samples for our studies were chosen using convenience, purposive and snowball sampling. Research studies conducted with conveniently selected sample suffer from the common problem of representativeness. As such the representativeness of our samples is a concern. Although we found some interesting results, we cannot claim that our findings represent the views of the wider population.

12.4.2 Internal Validity

We cannot guarantee that the survey questionnaires were interpreted by the participants in the way we wanted those to be. The development of survey questionnaires was carefully carried out in order to avoid the misinterpretation. However, our respondents came from different countries of the world and English is not spoken as first language in many of those. For example in our first survey the questions on the characteristics of high performing software testers might had been interpreted differently by our respondents. However, the enthusiastic and to-the-point responses to the open ended questions in most of our surveys give us confidence that questions were interpreted correctly by the participants in most of the cases.

There is a possibility that our participants provided random responses. This is a common threat to the type of empirical studies we conducted. However, we believe this did not happen in the studies we conducted. We do not see much motivation for the participants to provide random responses. One motivation could be the financial incentive that we offered in two of our studies (participants of the quasi experiment obtained $25 for participating and participant of software engineers' personality survey could register for a draw of two $100 Amazon vouchers). In the quasi experiment we noticed that the participants were enthusiastic and some of them suggested to us to increase the time for participation. In the survey of personality profiles, 46% registered for the draw of prize, so this was not primary motivation for many of the participants.

12.4.3 Construct Validity

For the assessment of effectiveness in software testing in our quasi experiment, we developed an instrument ourselves. The reliability of the instrument in measuring effectiveness in testing is a concern. The instrument needs to be verified with more experiments.

The appropriateness of the words used in survey questionnaire threatens the construct validity. The interpretation and often to some extent the responses depend on the ways questions are asked. We think this threat was present in our first survey. We learned from the experience and in the later surveys our careful design of questionnaire helped to mitigate this threat.

12.4.4 Conclusion Validity

In analysis of the qualitative data in most of our surveys, we applied grounded theory and content analysis. These are structured procedures used to interpret opinion-type responses. However, the interpretation can depend on the person analysing the data. To mitigate the influence of any bias most of the qualitative data was interpreted by the student researcher first and then checked by one of the supervisors.

12.5 Lessons Learned

A total of five research studies were conducted during this PhD. Four of those research studies were web-based surveys. One of those is reported as case study in Chapter 7 due to very low response rate. We refer to the studies as surveys here.

In conducting the four web-based surveys described, we came across several factors that we believe had an effect on our response rates and hence quantity and quality of data. We list some of the lessons we have learned from our experience of conducting the web based surveys in this section.

12.5.1 Participant Recruitment

Since all the surveys were conducted as part of a PhD thesis, the time to design, to get approval from the University human research ethics committee, to collect data, and to analyse the results were all limited. Hence we searched for a centralized body of software testing professionals where we could send invitations to participate in our survey in order to be able to recruit large numbers of participants in a relatively short time. However, there is no such centralized board of software testing professionals.

The relevant email lists provided by LinkedIn and Yahoo! were helpful in this regard. We believe this process of recruitment helped us to get a good number of participants in the short available time, for two of our surveys.

12.5.2 Low Response Rate

A crucial factor in survey research is the response rate, calculated as the ratio of the number of participants who completed the survey to the number of participants who we sent invitations to. A good survey aims to keep non-responses to a minimum. In the surveys (except the second survey) we conducted, it was impossible to calculate an accurate response rate, since most of the invitations were sent to the groups and the number of group members who actively read emails cannot be obtained. However, in comparison to the number of people we sent invitations to, the number of responses we received for the surveys is very poor. A common reason behind this low response could be that IT professionals may be reluctant to participate in research studies like a survey! However, this assumption, to our knowledge, has not been supported by any specific experimental evidence. Additionally, we found that academic paper referees are generally very keen to see specific response rates calculated and presented in papers using survey techniques. Being unable to calculate these due to recruitment techniques via boards and lists is not always well-received.

In conducting surveys, along with the survey response rate, the representativeness of the sample is also very important. According to Cook et al. [149], the assurance that the chosen sample represents the population is particularly important when a sample of convenience is chosen. In the surveys we conducted, the captured demographic information of the participants increased our confidence on the representativeness of the sample.

12.5.3 Invitation Email

We believe the nature of the invitation played an important role in recruiting participants for our surveys. We sent large invitation emails to the participants in the first three surveys. However, in the fourth survey, we sent a much shorter, catchy slogan and obtained far more participants compared to our other surveys.

12.5.4 Consent Collection

According to standard human research ethics protocols, participants are generally required to indicate their informed consent by signing Consent Information Statement (CIS) before participating. In our first, second and fourth surveys, participants could sign CIS electronically. However, in the third survey we requested participants to send a signed CIS to the researchers via email. We think the necessity of making two separate efforts (sending email and participating online) for participating might have negatively influenced participation.

Getting consent from the supervisor, manager or employer added an extra work burden on the participants. The reaction on getting manager consent was illustrated with the comment of one interested participant- "*...getting the consent from employer or manager is hard job and would prefer to avoid it. But if you need my opinion do send me the link, I can fill it up and can happily provide the information.*".

12.5.5 Questionnaire Length

In the series of surveys we conducted, we found that we obtained more respondents when the stated length of the survey was around 20-30 minutes. For the longer duration surveys, such as the worklog collection (at least 15 days), we got very few participants.

Figure 12.1 shows the number of responses compared to the length of the survey. The length of the second survey cannot be compared directly to the others since the time needed to complete a worklog might be only a few minutes (depending on the work load of the participants). However, participants were required to spend that few minutes each working day for at least 15 days. As such the exact time needed to complete the survey is unknown. Also the responses to the third survey are not conclusive, as due to the low number of responses, we stopped collecting data for the survey and modified the survey to make it less

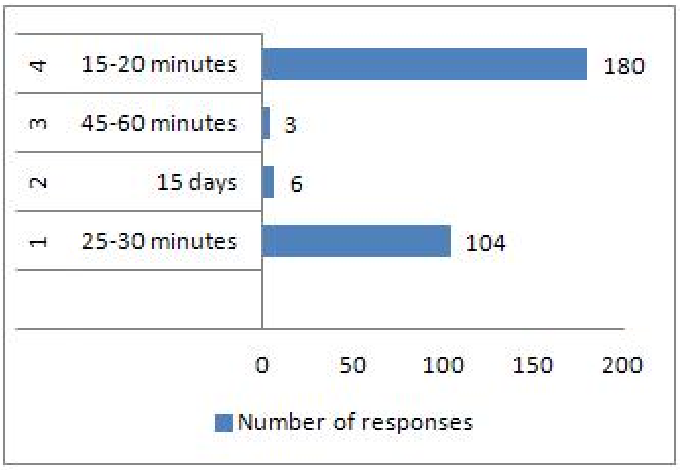


Figure 12.1: Number of responses compared to questionnaire length

time consuming.

12.5.6 Nature of Participation

We obtained a much higher number of participants in our first and fourth surveys. Compared to these two surveys the responses we obtained for our second and third surveys were very low (and very disappointing). One factor behind this might be the nature of participation requested in the surveys. The first and fourth surveys were prepared with a straightforward questionnaire. On the other hand, the second and third survey requested participation such that participants had to recall information from memory that took more time to form responses to questions. This may be a significant causative factor in the low response in those two surveys.

12.5.7 Motivation

We found that some IT professionals are highly motivated to participate in the surveys we conducted. The enthusiastic responses of the participants in the first survey encouraged us to design this series of surveys. In order to store the lengthy textual responses to open-ended questions in the first survey, we had to increase the size of our database field during data collection. Along with filling out the surveys, some members of Yahoo! groups, carried out discussion in the groups on the survey topic. We received the following response from one member to the group invitation indicating their interest "*Fascinating! It's an interesting study. I'd love to hear any conclusions you draw from this*". Members also informed their colleagues about the surveys without being explicitly asked to do so.

Some participants, on the other hand, participated for personal self-interest. For example, one participant was interested in postgraduate research and asked for help from the student researcher. The following comment is indicative: "*I am interested about SQA (Software Quality Assurance) and want to (do) research. I read your mail but i am not fully understand(ing) about the mail purpose. I want to participate in the research study about SQA (Software Quality Assurance). Can you help me?*".

12.5.8 Analysis of Data

The surveys were all originally conceived and designed to allow detailed quantitative analysis of the collected data. However, due to the limited responses received to our second and third studies, we were limited to performing qualitative analysis of the data instead. The qualitative analysis was very time-consuming although we did obtain useful and interesting results. However, had we known that we would need to use qualitative analysis, we would have asked some questions differently and some different questions.

12.5.9 Process of Ethics Approval

In the process of obtaining human research ethics committee approval, we had to submit a filled out protocol form accompanied by templates of the informed consent letters, website design documents and so on. The committee typically took a few weeks to review the application. We found the opinions of the committee members helped to improve the quality of the surveys. Delays were, however, compounded when it was necessary to make changes, often requiring another round of ethical review. Our new light-weight survey on software tester performance appraisal was an example of this.

12.5.10 Data Security

In web-based surveys, data usually is temporarily stored on a web server. The security of the data is a concern when those reside on an external server. Ensuring the security of the collected data was given higher priority in designing all the studies. This was also one of the primary concerns of the human research ethics committee that approved these experimental studies.

The data we collected were anonymous. Also considering the nature of the data, we believe even if the data would have been compromised, those would not be of any personal interest to others. In case of highly sensitive data like email addresses, we encrypted the data before storing. The University ethics committee appraisal of our applications was highly concerned to ensure very secure data management. We believe due to the higher importance given to the security issues during designing of the studies, we didn't face any problem.

12.6 Summary

In the series of research studies conducted we found that the effectiveness of software testing varies from tester to tester. There are number of factors that influence the effectiveness. Highly effective software testers exhibit some characteristics such as open mindedness. We also found that software testing includes variety of unit tasks and different criteria should be considered while assessing the performance of a person in carrying out these responsibilities. In a quasi experiment we found two major personality traits, extraversion and conscientiousness, are positively associated with effectiveness in software testing. In a broad survey we also found that software testers are higher on conscientiousness as compared to other software development partitioners. We discussed these findings in detail in this chapter.

We also described our experiences of conducting a series of web surveys on software testing-related topics. We found the most difficult part of the process is to convince participants. Although the group invitations helped to reach a large number of potential participants, we did not achieve a high response rate for two surveys. In the case of our third survey, we had to significantly modify the survey itself - and thus the research we were attempting to undertake - in order to get more responses!

We believe our experiences will be helpful for future researchers expecting to apply similar methodologies in similar fields. Based on our experiences we make the following recommendations to others who want or need to use surveying of IT professionals in their research programme -

1. Group invitations are helpful - but moderators need convincing of the worth.
2. Surveys designed with straightforward questionnaires that can be answered without too much thinking are more popular than surveys requiring more time to form responses - deeper research questions may massively reduce response and quality.
3. Marketing is important - a short, catchy slogan is more likely to draw the attention of participants.
4. Simple methods for participants to indicate informed consent are helpful in reducing drop-out rates.
5. Surveys should be designed with different data analysis methods in mind.
6. A significant amount of time should be allocated to the process of human research ethics approval.
7. In web surveys, the method of ensuring the security of data should be given a high priority.

Chapter 13

Conclusions

13.1 Introduction

This chapter summarizes the overall findings of the research, outlines the contributions made in context of academic research as well as industrial practice and proposes possible future work directions.

13.2 Research Summary

The thesis began with outlining different research questions related to software testers. The formal research questions were specified in later chapters. Four web based surveys and a quasi experiment were conducted to address these research questions.

In our first web based survey we found unit tasks of software testing from different sources of information including worklogs of software testers, job advertisements for software testers and bug reports of software testing related tools from open source bug repositories. We found that there was a number of testing related unit tasks performed by software testers. Those included creating test cases, preparing environment, executing test cases and so on. Apart from reporting test results testers perform different writing related tasks as well, such as preparing test documents, writing test status reports. Number of "research and development" type of tasks such as requirement analysis, research on testing tools and new technologies, preparing and improving quality standards are also performed by software testers. Other unit tasks such as debugging, planning, maintenance, managerial and collaboration with others are performed by testers.

In the preliminary web based survey we found that the performance of software testing varies depending on the individual. We also found number of bugs found is not a good metric to measure software testes' performance. The participants of this survey also suggested open mindedness is a characteristic of high performing software testers.

For selection of members to develop an effective software testing team, "Testing performance", "Experience in testing" and "Knowledge of problem domain" are the most important and "Training/certification in testing" and "Compatibility with other team members" are the least important factors to consider. "Diversity of professional background/experience" and "Diversity of personality" are also important for an effective software testing team. We also found that, a testing team performs better when they have experience working as a TEAM, rather than gathering experience as individuals.

We collected the personality profiles of software testers and related practitioners to identify similarities and dissimilarities. We found that compared to other software developers, software testers scored higher on conscientiousness.

In our quasi experimental study conducted with student participants to find out the effect of personality traits on the effectiveness in a custom designed software testing task, we found that extraversion was negatively correlated with overall testing effectiveness, conscientiousness was positively correlated with bug location rate and was negatively correlated with weighted fault density. None of the association were however, strong. In testing for hypothesis we found different level of conscientiousness had influence on effectiveness in software testing.

In our final research study we collected information on performance appraisal of software testers and feedback on our proposed performance appraisal form for software testers. We found most organizations practice specialized performance appraisal process for software testers, with a few organizations using common performance appraisal methods for all employees. We also identified a number of factors that are important in appraising testers. "Bug reported after delivery" and "efficiency of test case execution" were most prominent of those. We also refined our proposed PAF based on the feedback and we believe the new refined PAF can appraise software testers' performance appropriately.

13.3 Research Contributions

In this section, we list the contributions that were made in context of industry practice and academic research on software testing.

13.3.1 Software Testing Research

1. The enhanced knowledge on software testing unit tasks will help researchers in further investigation of factors that influence the effectiveness in this role.

2. The indication of the influence of different factors on the performance of software testers and the relative importance of different factors in measuring performance of software testers will help in designing more specific research studies to investigate the effect of each identified factors.

3. This research was aimed to find any connection between personality and effectiveness in software testing, however, preliminary comments on the influence of some other factors such as experience, automated tools indicates more research is needed on this.

4. The preliminary finding indicates open mindedness, extraversion and conscientiousness may influences software testing effectiveness. More specific and detailed research can now be designed to investigate the influence of each factor.

13.3.2 Software Testing Practice

1. We believe the identified unit tasks of software testing will help recruiters to design job responsibilities for testers.

2. The list of software testing unit tasks will also help young graduates to get better understanding of the responsibilities of this role in selection of their career choices.

3. From the preliminary survey we found open mindedness is believed by practitioners to be a characteristic of high performing software testers, and from our quasi experiment we found conscientiousness and extraversion have some connection with effectiveness in software testing. This will help employers to recruit software testers.

4. The personality traits found to be influential to software testers' performance can also help young IT graduates select an appropriate -career path for them.

5. Our proposed Performance Appraisal Form (PAF) can be adopted by organizations to appraise performance the software testers.

6. The information and suggestion collected on the performance appraisal of software testers we believe will help organizations to improve their performance appraisal process.

13.4 Limitations

A potential limitation of our research was the small sample size. Due to the sample size we could not strongly conclude any of the findings. We could not apply statistical tests in all our research studies due to this reason.

In absence of a well accepted standard for assessment of software testing performance, we used a small software testing task designed by us in the quasi experiment. The efficiency of this task was not verified with any prior research study.

The majority of our survey was designed to collect and analyse qualitative data. A potential drawback of dealing with qualitative data is that the analysis and interpretation of such data is greatly dependent on the researchers.

13.5 Future Research

All of the research studies conducted as part of the thesis need to be replicated with larger sample to validate the finding. The research reported in this thesis also opened window for more research in to software testing. Along with personality the influence of other human factors on software testing also need to be investigated with more detailed research. Some specific plan for our future work is presented here:

13.5.1 Validation of Proposed Performance Appraisal Form

We plan to validate our proposed performance appraisal form by requesting IT organizations to use the form to appraise their software testers. The same software testers will also be appraised with their existing appraisal process. The appraisal obtained using both methods will then be compared to confirm the effectiveness of our proposed PAF to appraise software testers appropriately.

13.5.2 Influence of Personality Traits on Software Testers' Performance

We propose to design a research study with professional software testers to investigate the association between personality traits and performance. We aim to employ our proposed performance appraisal form to assess performance of software testers, once the form is validated. We will collect the personality profiles (big five personality traits) of the software testers and will analyze the personality profiles and performance of software testers to find association, if there is any.

13.5.3 Influence of Conscientiousness on Effectiveness in Software Testing

From our research studies we found that conscientiousness was associated with bug location rate and weighted fault density of student software testers. We also found software testers working in the industry were higher on conscientiousness compared to other software development practitioners. Overall, there are some indication that conscientiousness might be influencial to software testers' performance. We plan to investigate the influence of conscientiousness on software testing performance with more specific research study. In the proposed research study the effect of other personality traits will be kept to a minimum. Groups of software testers will be formed based on the score on conscientiousness. The performance in software testing of each group will be assessed and compared.

13.5.4 Detailed Study of Software Testers' Unit Tasks

In our research study investigating unit tasks of software testers, we captured worklog of software testers, collected job advertisements for software testers and extracted bug reports to prepare a list of job responsibilities of software testers. However, the number of job advertisements and bug reports considered were small. In future, we plan to conduct research study mining more bug repositories and analyzing larger number of job advertisements. Also we plan to adopt different methodology for collecting worklog such as ethnography using think aloud approach.

13.5.5 Association of Software Testing Unit Tasks and Personality Traits

With the list of the unit tasks performed by software testers, we want to conduct a research study to find what personality traits are helpful to successfully carry out what unit task.

13.6 Summary

Due to the nature of the role of software testers, it is assumed that their personality traits can be different to other software developers and certain personality traits can influence their performance in this role. The research conducted to find such traits revealed that software testers were more conscientious than other software developers. We also found that conscientiousness and extraversion have some influence on the performance of software testers. These finding will help researchers to formulate hypothesis and design further research to investigate the effect of conscientiousness on software testing. This will also help practitioners to recruit software testers as well as young pupil in their career selection.

References

[1] P. T. Costa and R. R. McCrae, *The NEO personality inventory manual*, Psychological Assessment Ressources, Odessa, FL, 1985.

[2] D. Grote, *The Complete Guide to Performance Appraisal.* Amacom Books, 1996. [Online]. Available: http://books.google.com.au/books?id=u1n3tgAACAAJ

[3] redOrbit Staff & Wire Reports, "Software testing market continues to rise." [Online]. Available: http://www.redorbit.com/news/technology/1652726/software_testing_market_continues_to_rise/

[4] A. Parthasarathy, "Testing times ahead? its good news for india!" January 20, 2008.

[5] A. Abran, P. Bourque, R. Dupuis, J. W. Moore, and L. L. Tripp, *Guide to the Software Engineering Body of Knowledge - SWEBOK*, 2004th ed., A. Abran, P. Bourque, R. Dupuis, J. W. Moore, and L. L. Tripp, Eds. Piscataway, NJ, USA: IEEE Press, 2004. [Online]. Available: http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf

[6] G. M. Weinberg, *The Psychology of Computer Programming.* New York, NY, USA: John Wiley & Sons, Inc., 1985.

[7] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed. McGraw-Hill Higher Education, 2005.

[8] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," in *FOSE '07: 2007 Future of Software Engineering.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 85–103.

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

[10] P. G. Armour, "The Unconscious Art of Software Testing," *Communications of the ACM*, vol. 48, no. 1, pp. 15–18, 2005.

[11] J. Iivonen, M. V. Mäntylä, and J. Itkonen, "Characteristics of high performing testers: a case study," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '10. New York, NY, USA: ACM, 2010, pp. 60:1–60:1.

[12] L. Shoaib, A. Nadeem, and A. Akbar, "An Empirical Evaluation of The Influence of Human Personality on Exploratory Software Testing," in *Multitopic Conference, 2009. INMIC 2009. IEEE 13th International*, 2009, pp. 1 –6.

[13] A. Beer and R. Ramler, "The Role of Experience in Software Testing Practice," in *SEAA '08: Proceedings of the 2008 34th Euromicro Conference*

Software Engineering and Advanced Applications. Washington, DC, USA: IEEE Computer Society, 2008, pp. 258–265.

[14] J. Itkonen, M. V. Mantyla, and C. Lassenius, "How do testers do it? an exploratory study on manual testing practices," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 494–497. [Online]. Available: http://dx.doi.org/10.1109/ESEM.2009.5314240

[15] L. F. Capretz and F. Ahmed, "Making Sense of Software Development and Personality Types," *IT Professional*, vol. 12, pp. 6–13, 2010.

[16] ——, "Why Do We Need Personality Diversity in Software Engineering?" *ACM SIGSOFT Software Engineering Notes*, vol. 35, pp. 1–11, March 2010.

[17] H. Shah and M. J. Harrold, "Studying Human and Social Aspects of Testing in a Service-Based Software Company: Case Study," in *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '10. New York, NY, USA: ACM, 2010, pp. 102–108. [Online]. Available: http://doi.acm.org/10.1145/1833310.1833327

[18] J. Rooksby, M. Rouncefield, and I. Sommerville, "Testing in the Wild: The Social and Organisational Dimensions of Real World Practice," *Comput. Supported Coop. Work*, vol. 18, pp. 559–580, December 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1666305.1666323

[19] C. Kaner, “Measuring the effectiveness of software testers,” *Software Testing Analysis Review Conference (STAR) East*, May 2003.

[20] A. M. Turing, “Computing machinery and intelligence,” pp. 433–460, 1950. [Online]. Available: http://cogprints.org/499/

[21] G. J. Myers and C. Sandler, *The Art of Software Testing.* John Wiley & Sons, 2004.

[22] D. Gelperin and B. Hetzel, “The growth of software testing,” *Commun. ACM*, vol. 31, pp. 687–695, June 1988. [Online]. Available: http://doi.acm.org/10.1145/62959.62965

[23] C. L. Baker, “Review of D.D.McCracken’s “Digital Computer Programming”,” 1957.

[24] I. Burnstein, A. Homyen, T. Suwanassart, G. Saxena, and R. Grom, “A testing maturity model for software test process assessment and improvement,” *Software Quality Professional*, vol. 1, no. 4, pp. 8–21, 1999.

[25] B. Beizer, *Software Testing Techniques (2nd ed.).* New York, NY, USA: Van Nostrand Reinhold Co., 1990.

[26] E. J. Weyuker, T. J. Ostrand, J. Brophy, and R. Prasad, “Clearing a Career Path for Software Testers,” *IEEE Softw.*, vol. 17, pp. 76–82, March 2000. [Online]. Available: http://portal.acm.org/citation.cfm?id=624636.626113

[27] T. Kasse, *Practical Insight into CMMI*, 2nd ed. Norwood, MA, USA: Artech House, Inc., 2008.

[28] "MBTI," http://www.personalitypathways.com/MBTI_intro.html.

[29] B. Pettichord, "Testers and Developers Think Differently: Understanding and Utilizing the Diverse Traits of Key Players on Your Team," *Testing & Quality*, vol. 2, January-February 2000.

[30] M. Pol, R. Teunissen, and E. V. Veenendaal, *Software Testing:A guide to the TMap Approach.* Addison-Wesley (E), 2001.

[31] R. Black, "Being a "good" tester: Attitudes, skills, and growth," http://www.rbcs-us.com/documents/BeingaGoodTester.pdf.

[32] J.-F. Collard and I. Burnstein, *Practical Software Testing.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[33] A. D. da Cunha and D. Greathead, "Does Personality Matter? An Analysis of Code-Review Ability," *Communications of the ACM*, vol. 50, no. 5, pp. 109–112, May 2007.

[34] H. Almodaimeegh and J. Harrold, "Predicting Debugging Successs: An Investigation of the Relationship between Learning Styles, Personality Traits, and Computer Program Debugging ," in *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2009*, G. Siemens and C. Fulford, Eds. Honolulu, HI, USA: AACE, June 2009, pp. 2702–2710. [Online]. Available: http://www.editlib.org/p/31860

[35] J. B. Rotter, "Generalized expectancies for internal versus external control

of reinforcement." *Psychological monographs*, vol. 80, no. 1, pp. 1–28, 1966. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/5340840

[36] J. Bach, *Exploratory Testing Explained*, E. v. Veenendaal, Ed. The Testing Practitioner. UTN Publishers, 2002, www.satisfie.com.

[37] M. Rehman, A. Mahmood, R. Salleh, and A. Amin, "Mapping job requirements of software engineers to big five personality traits," in *Computer Information Science (ICCIS), 2012 International Conference on*, vol. 2, June, pp. 1115–1122.

[38] V. C. Petersen, "MBTI - Distorted Reflections of Personality?" University of Aarhus, Aarhus School of Business, Department of Management, Working Papers 2006-5, Sep. 2006. [Online]. Available: http://ideas.repec.org/p/hhb/aardom/2006_005.html

[39] G. J. Boyle, "Myers-Briggs Type Indicator (MBTI): Some Psychometric Limitations," *Australian Journal of Psychology*, vol. 30, 1995.

[40] R. A. Bjork and D. Druckman, *In the Mind's Eye: Enhancing Human Performance.* The natinal Academy Press, 1991.

[41] M. L. Lyons, "The DP Psyche," *Datamation*, vol. 31, no. 16, pp. 103–105, 1985.

[42] J. Chandler, J. Carter, and I. Benest, "Extrovert or introvert? the real personalities of computing students," in *Proceedings of 4th Annual*

LTSN-ICS conference. Galway: LTSN-ICS, August 2003. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2003/1678

[43] C. G. Cegielski and D. J. Hall, “What Makes a Good Programmer?” *Commun. ACM*, vol. 49, pp. 73–75, October 2006. [Online]. Available: http://doi.acm.org/10.1145/1164394.1164397

[44] D. P. Darcy and M. J. Ma, “Exploring Individual Characteristics and Programming Performance: Implications for Programmer Selection,” *Hawaii International Conference on System Sciences*, vol. 9, p. 314a, 2005.

[45] K. U. L. Arockiam, T. Lucia Agnes Beena and H. Leena, “Object-Oriented Program Comprehension and Personality Traits,” in *Proceedings of SMEF*, 2005.

[46] D. Greathead, “MBTI Personality Type and Student Code Comprehension Skill,” www.ppig.org, Tech. Rep., 2008.

[47] F. Ahmed, P. Campbell, A. Jaffar, S. Alkobaisi, and J. Campbell, “Learning & personality types: A case study of a software design course,” *Journal of Information Technology Education: Innovations in Practice*, vol. 9, no. 1, pp. 237–252, January 2010. [Online]. Available: http://www.editlib.org/p/111706

[48] L. F. Capretz, “Personality Types in Software Engineering,” *Int. J. Hum.-Comput. Stud.*, vol. 58, no. 2, pp. 207–214, 2003.

[49] R. Sach, M. Petre, and H. Sharp, "The use of MBTI in software engineering," in *22nd Annual Psychology of Programming Interest Group 2010*, September 2010. [Online]. Available: http://oro.open.ac.uk/24433/

[50] S. Cruz, F. da Silva, C. Monteiro, C. Santos, and M. dos Santos, "Personality in software engineering: Preliminary findings from a systematic literature review," in *Evaluation Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*, 2011, pp. 1–10.

[51] A. S. Sodiya, H. Longe, S. Onashoga, O. Awodele, and L. Omotosho, "An improved assessment of personality traits in software engineering," *Interdisciplinary Journal of Information, Knowledge and Management*, pp. 163–177, January 2007.

[52] R. Feldt, L. Angelis, R. Torkar, and M. Samuelsson, "Links between The Personalities, Views and Attitudes of Software Engineers," *Inf. Softw. Technol.*, vol. 52, pp. 611–624, June 2010. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2010.01.001

[53] J. G. Clark, D. B. Walz, and J. L. Wynekoop, "Identifying Exceptional Application Software Developers: A Comparison of Students and Professionals," *Communications of the Association for Information Systems*, vol. 11, 2003.

[54] J. Karn and A. Cowling, "A Study of The Effect of Disruptions on The Performance of Software Engineering Teams," *Empirical Software Engineering, International Symposium on*, vol. 0, p. 9 pp., 2005.

[55] J. Karn and T. Cowling, “A follow up study of the effect of personality on the performance of software engineering teams,” in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, ser. ISESE ’06. New York, NY, USA: ACM, 2006, pp. 232–241. [Online]. Available: http://doi.acm.org/10.1145/1159733.1159769

[56] V. Pieterse, D. G. Kourie, and I. P. Sonnekus, “Software Engineering Team Diversity and Performance,” in *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, ser. SAICSIT ’06. , Republic of South Africa: South African Institute for Computer Scientists and Information Technologists, 2006, pp. 180–186. [Online]. Available: http://dx.doi.org/10.1145/1216262.1216282

[57] R. H. Rutherfoord, “Using Personality Inventories to Form Teams for Class Projects: a Case Study,” in *Proceedings of the 7th conference on Information technology education*, ser. SIGITE ’06. New York, NY, USA: ACM, 2006, pp. 9–14. [Online]. Available: http://doi.acm.org/10.1145/1168812.1168817

[58] N. Gorla and Y. W. Lam, “Who Should Work with Whom?: Building Effective Software Project Teams,” *Commun. ACM*, vol. 47, pp. 79–82, June 2004. [Online]. Available: http://doi.acm.org/10.1145/990680.990684

[59] A. R. Peslak, “The Impact of Personality on Information Technology Team Projects,” in *Proceedings of the 2006 ACM SIGMIS CPR conference*

on computer personnel research: Forty four years of computer personnel research: achievements, challenges & the future, ser. SIGMIS CPR '06. New York, NY, USA: ACM, 2006, pp. 273–279. [Online]. Available: http://doi.acm.org/10.1145/1125170.1125233

[60] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *Journal of Systems and Software*, vol. 82, no. 11, pp. 1869 – 1890, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016412120900123X

[61] D. Martin, J. Rooksby, M. Rouncefield, and I. Sommerville, ""Good" Organisational Reasons for "Bad" Software Testing: An Ethnographic Study of Testing in a Small Software Company," in *Proceedings of the 29th international conference on Software Engineering*, ser. ICSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 602–611. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2007.1

[62] C. F. Cohen, S. J. Birkin, M. J. Garfield, and H. W. Webb, "Managing conflict in software testing," *Commun. ACM*, vol. 47, pp. 76–81, January 2004. [Online]. Available: http://doi.acm.org/10.1145/962081.962083

[63] F. Ahmed, L. Capretz, and P. Campbell, "Evaluating the demand for soft skills in software development," *IT Professional*, vol. 14, no. 1, pp. 44–49, Jan.-Feb.

[64] N. Fenton and S. L. Pfleeger, *Software metrics (2nd ed.): a rigorous and practical approach.* Boston, MA, USA: PWS Publishing Co., 1997.

[65] R. B. Grady and D. L. Caswell, *Software metrics: establishing a company-wide program.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1987.

[66] C. Kaner, "Don't use bug counts to measure testers," *Software Testing & Quality Engineering*, p. 80, May/June 1999.

[67] B. L. Killingsworth, M. B. Hayden, D. Crawford, and R. Schellenberger, "A model for motivating and measuring quality performance in information systems staff," *Information Systems Management*, vol. 18, no. 2, pp. 1–7, 2001. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1201/1078/43195.18.2.20010301/31271.2

[68] D. B. Mayer and A. W. Stalnaker, "Selection and evaluation of computer personnel- the research history of sig/cpr," in *Proceedings of the 1968 23rd ACM national conference*, ser. ACM '68. New York, NY, USA: ACM, 1968, pp. 657–670. [Online]. Available: http://doi.acm.org/10.1145/800186.810630

[69] R. A. Dickmann, "A programmer appraisal instrument," in *Proceedings of the second SIGCPR conference on Computer personnel research*, ser. SIGCPR '64. New York, NY, USA: ACM, 1964, pp. 45–64. [Online]. Available: http://doi.acm.org/10.1145/1142635.1142640

[70] E. F. Bairdain, "Research studies of programmers and programming," 1964.

[71] R. M. Berger and R. C. Wilson, "Correlates of programmer proficiency," in *Proceedings of the fourth SIGCPR conference on Computer personnel research*, ser. SIGCPR '66. New York, NY, USA: ACM, 1966, pp. 83–95. [Online]. Available: http://doi.acm.org/10.1145/1142620.1142629

[72] B. Powell, "Performance evaluation of programmers and analysts," in *Proceedings of the 3rd annual ACM SIGUCCS conference on User services*, ser. SIGUCCS '75. New York, NY, USA: ACM, 1975, pp. 19–21. [Online]. Available: http://doi.acm.org/10.1145/800115.803716

[73] R. J. Larsen and D. M. Buss, *Personality Psychology: Domains of Knowledge About Human Nature*, 3rd ed. McGraw-Hill, 2008.

[74] R. Williams, *Keywords : a vocabulary of culture and society*, ser. Fontana communications series. Fontana/Croom Helm, 1976. [Online]. Available: http://books.google.com.au/books?id=UqkOAAAAQAAJ

[75] N. Haslam, *Introduction to Personality and Intelligence.* London, UK: SAGE Publications Ltd, 2007.

[76] D. P. McAdams, *The Person: An Integrated Introduction to Personality Psychology*, 3rd ed. Orlando, FL, USA: Harcourt, Inc, 2001.

[77] R. Lanyon and L. Goodstein, *Personality assessment*, ser. Wiley Series on Personality Processes. Wiley, 1997. [Online]. Available: http://books.google.com.au/books?id=gt19AAAAMAAJ

[78] R. B. Cattell, "The description of personality: basic traits resolved into clusters," *The Journal of Abnormal and Social Psychology*, vol. 38, pp. 476–506, 1943.

[79] D. Watson and L. A. Clark, "Negative affectivity: The disposition to experience aversive emotional states." *Psychological Bulletin*, vol. 96, no. 3, pp. 465–490, Nov. 1984.

[80] D. W. Fiske, "Consistency of the factorial structures of personality ratings from different sources," *The Journal of Abnormal and Social Psychology*, vol. 44, no. 3, pp. 329–344, Jul. 1949.

[81] J. M. Digman, "Personality structure: Emergence of the five-factor model," *Annual Review of Psychology*, vol. 41, no. 1, pp. 417–440, 1990. [Online]. Available: http://www.annualreviews.org/doi/abs/10.1146/annurev.ps.41.020190.002221

[82] H. J. Eysenck, "The scientific study of personality," *British Journal of Statistical Psychology*, vol. 6, no. 1, pp. 44–52, 1953. [Online]. Available: http://dx.doi.org/10.1111/j.2044-8317.1953.tb00132.x

[83] A. Tellegen, "Brief manual for the differential personality questionnaire," *University of Minnesota*, 1982.

[84] C. Dyer, *Research in Psychology: A Practical Guide to Methods and Statistics*. Wiley, 2006. [Online]. Available: http://books.google.com.au/books?id=5iYO4h-NLV4C

[85] R. Robins, R. Fraley, and R. Krueger, *Handbook of Research Methods in Personality Psychology.* Guilford Publications, 2009. [Online]. Available: http://books.google.com.au/books?id=-VTvN3aPw8sC

[86] I. Myers and P. Myers, *Gifts differing: understanding personality type.* Davies-Black Pub., 1980. [Online]. Available: http://books.google.com.au/books?id=D3V-AAAAMAAJ

[87] "MBTI," http://members.ozemail.com.au/ alchymia/library/mb-tiorg.html.

[88] D. J. Pittenger, "Measuring the MBTI. . .And Coming Up Short," *Journal of Career Planning and Employment*, vol. 54, no. 1, pp. 48–52, November 1993.

[89] "SFPQ," http://www.sigmaassessmentsystems.com/resources/presentations/sfpq.pdf.

[90] I. Weiner and R. Greene, *Handbook of Personality Assessment.* , John Wiley & Sons, December 2007.

[91] L. R. Goldberg, *A broad-bandwidth, public-domain, personality inventory measuring the lower-level facets of several five-factor models.* Tilburg University Press, 1999, vol. 7, pp. 7–28. [Online]. Available: http://ipip.ori.org/New_IPIP-100-item-scale.htm

[92] L. R. Goldberg, J. A. Johnson, H. W. Eber, R. Hogan, M. C. Ashton, C. R. Cloninger, and H. G. Gough, "The international personality item pool and the future of public-domain personality measures," *Journal of Research in*

Personality, vol. 40, no. 1, pp. 84 – 96, 2006, proceedings of the 2005 Meeting of the Association of Research in Personality. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0092656605000553

[93] M. M. Harris and J. Schaubroeck, "A meta-analysis of self-supervisor, self-peer, and peer-supervisor ratings," *Personnel Psychology*, vol. 41, no. 1, pp. 43–62, 1988. [Online]. Available: http://dx.doi.org/10.1111/j.1744-6570.1988.tb00631.x

[94] P. A. Mabe and S. G. West, "Validity of self-evaluation of ability: A review and meta-analysis." *Journal of Applied Psychology*, vol. 67, no. 3, p. 280, 1982.

[95] W. E. Williams and D. A. Seiler, "Relationship between measures of effort and job performance." *Journal of Applied Psychology*, vol. 57, no. 1, p. 49, 1973.

[96] D. Pym and H. Auld, "The self-rating as a measure of employee satisfactoriness." *Occupational Psychology*, vol. 39, no. 2, pp. 103–113, 1965.

[97] R. J. Klimoski and M. London, "Role of the rater in performance appraisal." *Journal of Applied Psychology*, vol. 59, no. 4, p. 445, 1974.

[98] G. C. Thornton, "The relationship between supervisory- and self-appraisals of executive performance," *Personnel Psychology*, vol. 21, no. 4, pp. 441–455, 1968. [Online]. Available: http://dx.doi.org/10.1111/j.1744-6570.1968.tb02044.x

[99] J. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches.* SAGE Publications, 2003. [Online]. Available: http://books.google.com.au/books?id=nSVxmN2KWeYC

[100] M. Denscombe, *The Good Research Guide for Small-Scale Social Research Projects.* Milton Keynes, UK: Open University Press, 2003.

[101] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation.* Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[102] "IQNITE," http://www.iqnite-conferences.com/AU/index.aspx.

[103] F. Shull, J. Singer, and D. I. Sjøberg, *Guide to Advanced Empirical Software Engineering.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[104] S. Boslaugh and P. A. Watters, *Statistics in a Nutshell.* Sebastopol, CA, USA: O'Reilly, 2008.

[105] L. L. Tripp, "Software Certification Debate: Benefits of Certification," *Computer*, vol. 35, pp. 31–33, 2002.

[106] A. Kolawa, "Software Certification Debate: Certification Will Do More Harm than Good," *Computer*, vol. 35, pp. 34–35, 2002.

[107] "ISTQB homepage," http://www.istqb.org/.

[108] S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen, "A Preliminary Survey on Software Testing Practices in Australia," in *Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04)*, 2004, pp. 116–125.

[109] S. G. Cohen and D. E. Bailey, “What makes teams work: Group effectiveness research from the shop floor to the executive suite,” *Journal of Management*, vol. 23, no. 3, pp. 239–290, 1997. [Online]. Available: http://jom.sagepub.com/content/23/3/239.abstract

[110] J. H. Bradley and F. J. Hebert, “The effect of personality type on team performance,” *The Journal of Management Development*, vol. 16, no. 5, pp. 337–353, 1997. [Online]. Available: http://dx.doi.org/10.1108/02621719710174525

[111] S. Faraj and L. Sproull, “Coordinating expertise in software development teams,” *Manage. Sci.*, vol. 46, pp. 1554–1568, December 2000. [Online]. Available: http://portal.acm.org/citation.cfm?id=970301.970322

[112] S. K. Horwitz and I. B. Horwitz, “The Effects of Team Diversity on Team Outcomes: A Meta-Analytic Review of Team Demography,” *Journal of Management*, vol. 33, no. 6, pp. 987–1015, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1177/0149206307308587

[113] W. C. Schutz, *The Interpersonal Underworld*. Palo Alto, CA, USA: Science and Behavior Books, 1966.

[114] S. Cox and R. Osguthorpe, “How do instructional design professionals spend their time?” *TechTrends*, vol. 47, no. 3, pp. 45–47, 2003. [Online]. Available: http://dx.doi.org/10.1007/BF02763476

[115] “Triangulation,” http://en.wikipedia.org/wiki/Triangulation_social_science.

[116] "Recruitment website," http://www.monster.com/, [Online; accessed 07-July-2010].

[117] "Eclipse JDT," http://www.eclipse.org/jdt/.

[118] "Firefox Testopia," http://www.mozilla.org/projects/testopia/.

[119] I. Myers, *A Guide to the Development and Use of the Myers-Briggs Type Indicator: Manual.* Consulting Psychologists Press, 1986. [Online]. Available: http://books.google.com.au/books?id=YWXDHAAACAAJ

[120] R. R. McCrae and O. P. John, "An introduction to the five-factor model and its applications," *Journal of Personality*, vol. 60, no. 2, pp. 175–215, 1992. [Online]. Available: http://dx.doi.org/10.1111/j.1467-6494.1992.tb00970.x

[121] R. Merkel and T. Kanij, "Does the individual matter in software testing?" http://www.swinburne.edu.au/ict/research/sat/-technicalReports/TC2010-001.pdf.

[122] T. Kanij, R. Merkel, and J. Grundy, "A preliminary study on factors affecting software testing team performance," in *Empirical Software Engineering and Measurement*, 2011, pp. 359–362.

[123] L. F. Sanz, "Personal skills for computing professionals." *IEEE Computer*, no. 10, pp. 110–112.

[124] J. A. Johnson, "The IPIP NEO personality assessment tools," http://www.personal.psu.edu/j5j/IPIP/.

[125] N. Salleh, E. Mendes, and J. Grundy, "Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments," *Empirical Software Engineering*, pp. 1–39, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10664-012-9238-4

[126] J. A. Gliem and R. R. Gliem, "Calculating , interpreting , and reporting cronbach s alpha reliability coefficient for likert-type scales," *October*, vol. 88, no. 1992, pp. 82–88, 2003. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Calculating+,+Interpreting+,+and+Reporting+Cronbach++s+Alpha+Reliability+Coefficient+for+Likert-Type+Scales#0

[127] T. Buchanan, J. A. Johnson, and L. R. Goldberg, "Implementing a Five-Factor Personality Inventory for Use on the Internet," *European Journal of Psychological Assessment*, vol. 21, no. 2, pp. 115–127, 2005. [Online]. Available: file://D:%5CArticles%5CMeasures%5CBuchananGoldgerg2005FiveFactPersInvNet.pdf

[128] D. George and P. Mallery, *SPSS for Windows Step by Step: A Simple Guide and Reference.* Allyn and Bacon, 1999. [Online]. Available: http://books.google.com.au/books?id=7tYnAQAAIAAJ

[129] T. Buchanan, "Online implementation of an ipip five factor personality inventory," http://www.docstoc.com/docs/36825859/Big-Five—Online-Implementation-of-an-IPIP-Five-Factor, March 2013.

[130] R. A. Fisher, "On the probable error of a coefficient of correlation deduced from a small sample," *Metron*, vol. 1, pp. 3–32, 1921.

[131] A. Furnham, "The big five versus the big four: the relationship between the myers-briggs type indicator (MBTI) and NEO-PI five factor model of personality," *Personality and Individual Differences*, vol. 21, no. 2, pp. 303 – 307, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191886996000335

[132] P. T. Costa and R. R. McCrae, *Revised NEO Personality Inventory (NEO-PI R) and Neo Five Factor Inventory (NEO-FFI)*. Psychological Assessment Inventories, 1992.

[133] M. L. Hutcheson, *Software Testing Fundamentals: Methods and Metrics*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[134] "Bug triage meeting - severity & priority," http://geekswithblogs.net/srkprasad/archive/2004/08/20/9961.aspx.

[135] J. Ploski, M. Rohr, P. Schwenkenberg, and W. Hasselbring, "Research issues in software fault categorization," *SIGSOFT Softw. Eng. Notes*, vol. 32, November 2007. [Online]. Available: http://doi.acm.org/10.1145/1317471.1317478

[136] "EclipseMetrics," http://www.stateofflow.com/projects/16/eclipsemetrics.

[137] "IEEE Standard for Software Test Documentation," *IEEE Std 829-1998*, p. i, 1998.

[138] T. Kanij, R. Merkel, and J. Grundy, “Performance assessment metrics for software testers,” in *CHASE*, 2012, pp. 63–65.

[139] “Survey Monkey,” http://www.surveymonkey.com/.

[140] “Fluid surveys,” http://fluidsurveys.com/.

[141] “Free online surveys,” http://freeonlinesurveys.com/.

[142] “Kwik surveys,” http://kwiksurveys.com/.

[143] D. Wall, *Multi-Tier Application Programming with PHP: Practical Guide for Architects and Programmers*, ser. The Practical Guides. Elsevier Science, 2004. [Online]. Available: http://books.google.com.au/books?id=9xuuBm08bMwC

[144] “Hypertext markup language HTML,” http://en.wikipedia.org/wiki/HTML.

[145] “PHP: Hypertext preprocessor,” http://php.net/.

[146] “Javascript,” http://en.wikipedia.org/wiki/JavaScript.

[147] “Cascading style sheet css,” http://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[148] “MySQL,” http://www.mysql.com/.

[149] C. Cook, F. Heath, and R. L. Thompson, “A Meta-Analysis of Response Rates in Web- or Internet-Based Surveys,” *Educational and Psychological Measurement*, vol. 60, no. 6, pp. 821–836, December 2000.

Appdx A

Swinburne University of Technology

A survey of key factors affecting effectiveness of software testing professionals

Faculty of Information and Communication Technologies

Research Supervisor

Dr. Robert Merkel
Lecturer,
Faculty of Information and Communication Technologies
Swinburne University of Technology
PO Box 218
Hawthorn VIC 3122
Australia
Ph + 61392148741
Email:rmerkel@ict.swin.edu.au

Researcher

Tanjila Kanij

PhD Candidate
FICT, Swinburne University of Technology
Phone: +61 3 9214 8840
Email: tkanij@swin.edu.au
PO Box 218
Hawthorn VIC 3122
Australia
Ph + 61392148840
Email:tkanij@swin.edu.au

This survey invites the voluntary participation of software testing professionals. The aim of the survey is to identify the key factors that influence the workplace performance of software testers. This survey is conducted as a part of PhD research by Tanjila Kanij under the supervision of Dr. Robert Merkel, Lecturer, Faculty of ICT, Swinburne University of Technology. The project is approved by Swinburne University Human Research Ethics Sub Committee (SHESC2) and conducted under the privacy policy followed by Swinburne University of Technology. The responses provided will be kept confidential and will be used for academic research purposes only. Individual responses will not be released or shared. The information provided will be kept in our server and will be accessible to the researchers only. Results from analysis of survey responses will be published in peer-reviewed academic journals.

The survey questionnaire contains 29 questions taking at most 25 minutes to be answered. No identifying information is collected. If you wish to be informed of the results of the survey, you may provide an email address through which we will contact recipients with a report of the results once the study is completed; however, this address will be stored independently and NOT associated with your survey responses. You are also encouraged to ask if any questions you have about the survey. Your co-operation will be greatly helpful for the successful completion of this survey.

Swinburne University of Technology

The Effect of Personality Traits on Effectiveness of Software Testing

Faculty of Information and Communication Technologies

Research Supervisor

Prof. John Grundy

Professor in Software Engineering and Head of Academic Group, Computer Science & Software Engineering
FICT, Swinburne University of Technology
Phone: +61 3 9214 8731
Email: jgrundy@swin.edu.au

Research Supervisor

Dr. Robert Merkel

Lecturer
Clayton School of IT, Monash University
Phone: +61 3 9905 5056
Email: robert.merkel@monash.edu

Researcher

Tanjila Kanij

PhD Candidate
FICT, Swinburne University of Technology
Phone: +61 3 9214 8840
Email: tkanij@swin.edu.au

Dear Research Participant,

You are invited to participate in a research trial as part of PhD study by Tanjila Kanij, supervised by Prof. John Grundy and Dr. Robert Merkel in the Faculty of ICT of Swinburne University of Technology. The research project will examine the relationship between personality traits and effectiveness in software testing. Being a student of Faculty of ICT of Swinburne University of Technology, you are cordially requested to read the following description of the research project and if you are interested to participate in the project, sign this consent form before participating.

The Research Project

Software testing is a procedure of exploring software to find defects prior release. Our previous research has suggested that the psychological traits of software testers may influence their effectiveness in software testing. This research project is designed to examine the influence directly.

If you have any general inquiry about the research project, you are welcome to contact any of the researchers available at above mentioned addresses. This research project is approved on behalf of Swinburne's Human Research Ethics Committee (SUHREC) by a SUHREC Subcommittee (SHESC2). If you have any complaints or question regarding the approval of the project you can contact the ethics officer at resethics@swin.edu.au.

The Procedure

The research project is divided in two tests. If you agree to participate in the test, you will be requested to fill a NEO PI-R personality assessment form containing 240 multiple choice questions, in the first test. It will take around 40 minutes. **NEO PI 3 is a standard, commonly used test to assess personality trait. It is NOT a test to reveal psychological disorders.**

In the second test, you will be given a small software program to test and write a report about the errors you find in it. In this test, you will be provided a computer where a small program written in Java will be installed. A printed specification of the software program will be available.

The total session will take about 1.5 hours.

The data of the two tests will be analysed to see if there is a connection between personality traits and effective software testing. **No identifying information about you will be recorded; each participant will be assigned a unique code.** No individual results will be reported in a way that any participant can be indentified.

Expected Outcome

If specific personality traits having influence on the effectiveness of software testing can be formed, we can suggest the traits to be looked for while recruiting software testers. This will help the recruitment process of this profession which still relies on the experience and judgement of the recruiter. This will also help IT graduates to select appropriate IT career for them.

The outcome will be published as a technical report as soon as prepared and later in peer reviewed conferences and/or journals.

Participation

You will get $25 for participating in the research project. You will also get a copy of your personality profile called "My NEO Summary". You are free to withdraw at any time during the tests, if you feel uncomfortable. The researchers will be physically available during both the tests and will be happy to assist you if you have any question or need further explanation. If you are worried about the outcome of your personality assessment test, you are welcome to have discussion with the researchers. In the unlikely event if you find the personality test distressing, you can ask for counselling. If you wish to be informed about the research outcome you can optionally put your email address where a technical report on the research project will be sent as soon as it is available.

Consent

By signing this consent form, I confirm that I have read and understood the information provided about the research project and I agree to take part in this research project.

Print Full Name: ____

Signature ____

Date ____

Email (Optional, If you wish to receive a copy of the research outcome)____

SWINBURNE UNIVERSITY OF TECHNOLOGY

Swinburne University of Technology

What Do Software Testers Spend Their Time Doing?

Faculty of Information and Communication Technologies

Research Supervisor

Prof. John Grundy

Professor in Software Engineering and Head of Academic Group, Computer Science & Software Engineering
FICT, Swinburne University of Technology
Phone: +61 3 9214 8731
Email: jgrundy@swin.edu.au

Research Supervisor

Dr. Robert Merkel

Lecturer
Clayton School of IT, Monash University
Phone: +61 3 9905 5056
Email: robert.merkel@monash.edu

Researcher

Tanjila Kanij

PhD Candidate
FICT, Swinburne University of Technology
Phone: +61 3 9214 8840
Email: tkanij@swin.edu.au

This survey is part of a PhD research at Swinburne University of Technology.

The purpose of the survey is to collect the work log of software testers to revise a list of actual job responsibilities of software testers. The list of jobs accomplished by software testers will help the researchers for their future research as well as will help the employers to design job responsibilities for software testers.

If you are interested to participate please create an account to provide your work log for a minimum of two weeks.

Anonymity
Please note that a valid email address is required to create an account. The email address will not be associated with your user id. So a participant will NOT be identifiable with the user name. However, if you have any concerns you are advised to open a new "disposable" email account (using a free email service such as Google Mail), for this purpose. No identifying information of your employer or any of the other participants you worked with will be asked. Individual responses will not be shared with the employer company. However, the employer company may receive a copy of the outcome of the survey, if they wish to.

Confidentiality
The responses provided will be kept confidential and will be used for academic research purposes only. Individual responses will not be released or shared. The information provided will be kept in our server and will be accessible to the researchers only. Results from analysis of survey responses will be published in peer-reviewed academic conferences and journals.

Research Ethics
The project is approved by Swinburne University Human Research Ethics Sub Committee (SHESC2) and conducted under the privacy policy followed by Swinburne University of Technology.

Right to ask question
If you have any general inquiry about the research project, you are welcome to contact student researcher Tanjila Kanij at tkanij@swin.edu.au. If you have any complaints or question regarding the approval of the project you can contact with - Research Ethics Officer, Swinburne Research (H68), Swinburne University of Technology, P O Box 218, HAWTHORN VIC 3122. Tel (03) 9214 5218 or +61 3 9214 5218 or resethics@swin.edu.au

Voluntary Participation
Your participation in this research is voluntary. You may discontinue participation at any time during the research activity. You must check the consent box to indicate your consent to these terms.

SWINBURNE UNIVERSITY OF TECHNOLOGY

.

Swinburne University of Technology

Performance Assessment of Software Testers

Faculty of Information and Communication Technologies

Research Supervisor

Prof. John Grundy

Professor in Software Engineering and Head of Academic Group, Computer Science & Software Engineering
FICT, Swinburne University of Technology
Phone: +61 3 9214 8731
Email: jgrundy@swin.edu.au

Research Supervisor

Dr. Robert Merkel

Lecturer
Clayton School of IT, Monash University
Phone: +61 3 9905 5056
Email: robert.merkel@monash.edu

Researcher

Tanjila Kanij

PhD Candidate
FICT, Swinburne University of Technology
Phone: +61 3 9214 8840
Email: tkanij@swin.edu.au

Dear manager,

You are invited to participate in a research trial as part of PhD study by Tanjila Kanij, supervised by Prof. John Grundy and Dr. Robert Merkel in the Faculty of ICT of Swinburne University of Technology.

The research Study

The delivered reliability of software to large extend depends on the performance of software testers. Thus, appraising performance of software testers appropriately is crucial for ensuring quality project outcomes. However, to date there is very limited information available on how the appraisal of software testers is actually carried out in industry. More specifically, it is unknown what are the range of performance appraisal methods used, what are the pros and cons of the different methods, and are different methods contemporary or needing improvement.

This study is designed to collect information about the current practice of performance appraisal of software testers in industry. It will also evaluate a software tester performance appraisal form proposed by the researchers. This is based on our review of the literature and an analysis of job requirements for this position collected from recruitment advertisements.

The Procedure

The study is divided into two sections. In the first section the participants are requested to fill out a small survey questionnaire on the current practice of performance appraisal of testers in their organisation. The section contains a total of 12 questions. In the second section the participants are requested to consider one of their testers and to assign an overall performance appraisal score for that tester. The participants will then be given a performance appraisal form designed by the researchers and be asked to appraise the performance of that same tester using this form. The study will correlate the overall score assigned by the participants with the overall score calculated with the scores in the form. The participants are also asked to provide us their feedback on the form, positive or negative.

Anonymity
No identifying information about the participant will be recorded. Each participant will be assigned a unique code. No individual results will be reported in a way that any participant or their selected tester can be indentified.

No identifying information about your employer or any of the other people the participant has worked with will be asked. Individual responses will not be shared with the employer

SWINBURNE SWINBURNE UNIVERSITY OF TECHNOLOGY company. However, the employer company may receive a copy of the overall findings, if they wish to.

Confidentiality

The responses provided will be kept confidential and will be used for academic research purposes only. Individual responses will not be released or shared. Results from the analysis of responses will be published in a publically available PhD thesis, and may be published in peer-reviewed academic conferences and journals.

Research Ethics

This project has been approved by Swinburne's Human Research Ethics Subcommittee (SHESC3) on behalf of Swinburne's Human Research Ethics Committee (SUHREC) in line with the *National Statement on Ethical Conduct in Human Research (SUHREC Project 2011/172)*.

Right to ask question

If you have any general inquiry about the research project, you are welcome to contact the principal investigator Prof. John Grundy at jgrundy@swin.edu.au or the student researcher Tanjila Kanij at tkanij@swin.edu.au.

If you have any concerns or complaints about the conduct of this project, you can contact:

Research Ethics Officer, Swinburne Research (H68),
Swinburne University of Technology, P O Box 218,
HAWTHORN VIC 3122.
Tel (03) 9214 5218 or +61 3 9214 5218 or resethics@swin.edu.au.

Voluntary Participation

Your participation in this research is voluntary. You may discontinue participation at any time during the research activity and, on request, will delete data collected from you (until publication, at which point deletion is no longer possible).

Consent and Authorization

To participate, you must sign the consent form below. Furthermore, a person in your organization with sufficient authority to permit your participation must sign the authorization form. If you have the authority to authorize your own participation, you may sign the authorization form yourself.

Participant Consent

By signing this consent form, I confirm that I have read and understood the information provided about the research project and I agree to take part in this research project.

Print Full Name: ______________________________

Signature ______________________________

Date ______________________________

Email (Optional, If you wish to receive a copy of the research outcome)______________________________

Swinburne University of Technology

Personality Traits of Software Testers

Faculty of Information and Communication Technologies

Research Supervisor

Prof. John Grundy

Professor in Software Engineering and Head of Academic Group, Computer Science & Software Engineering
FICT, Swinburne University of Technology
Phone: +61 3 9214 8731
Email: jgrundy@swin.edu.au

Research Supervisor

Dr. Robert Merkel

Lecturer
Clayton School of IT, Monash University
Phone: +61 3 9905 5056
Email: robert.merkel@monash.edu

Researcher

Tanjila Kanij

PhD Candidate
FICT, Swinburne University of Technology
Phone: +61 3 9214 8840
Email: tkanij@swin.edu.au

You are invited to participate in a research trial as part of PhD study by Tanjila Kanij, supervised by Prof. John Grundy and Dr. Robert Merkel in the Faculty of ICT of Swinburne University of Technology.

The research Study

A software tester is a person who tests software before release, helping to increase the reliability of software products by reporting bugs and getting them fixed. This role is different to other stakeholders involved in the software development process. While software programmers are largely constructive, in that they design and "build" something that meets customer requirements, testers' job is in a sense destructive in that they attempt to "break" the software constructed by programmers. This needs a destructive approach towards the module under test. The alternative mindset required by testers poses a research question, whether software testers, as a group, tend to exhibit certain personality traits. This research study is designed to collect personality profiles of software testers to investigate whether certain personality traits are over-represented among software testers.

The Procedure

The study is conducted using an online questionnaire. Once participants give consent to participate in the study they will be presented with a small demographic questionnaire. On completion of the demographic questionnaire the participants will then complete the personality assessment questionnaire. **This section contains 50 statements describing a person. The participants are requested to select one response from 5 available responses ("Very Inaccurate", "Moderately Inaccurate", "Neither Inaccurate nor Accurate", "Moderately Accurate" and "Very Accurate") for each of the statements.** On submission of the responses the online system will return a general personality profile of the participant. Participants can also download this profile in PDF format.

Anonymity

No identifying information about participants will be recorded. Each participant will be assigned a unique code. No individual results will be reported in a way that any participant can be indentified.

No identifying information about your employer or any of the other people the participant has worked with will be asked. Individual responses will not be shared.

Confidentiality

The responses provided will be kept confidential and will be used for academic research purposes only. Individual responses will not be released or shared. Results from the analysis of responses will be published in a publically available PhD thesis, and may be published in peer-reviewed academic conferences and journals.

Research Ethics

This project has been approved by Swinburne's Human Research Ethics Subcommittee (<subcommittee name>) on behalf of Swinburne's Human Research Ethics Committee (SUHREC) in line with the *National Statement on Ethical Conduct in Human Research (SUHREC Project <project number>)*.

Right to ask question

If you have any general inquiry about the research project, you are welcome to contact the principal investigator Prof. John Grundy at jgrundy@swin.edu.au or the student researcher Tanjila Kanij at tkanij@swin.edu.au.

If you have any concerns or complaints about the conduct of this project, you can contact:

Research Ethics Officer, Swinburne Research (H68),
Swinburne University of Technology, P O Box 218,
HAWTHORN VIC 3122.
Tel (03) 9214 5218 or +61 3 9214 5218 or resethics@swin.edu.au.

Participation

Participants will be given the chance to enter a random draw to win one of two 100 USD Amazon gift vouchers for participating in the research projects. You need to enter your email address at the end of the **survey** to register for the draw of the incentive. No association between your email address and the responses to the survey questionnaire will be recorded or stored. You will also get a copy of your personality profile. You are free to withdraw at any time during the test, if you feel uncomfortable. If you are worried about the outcome of your personality assessment test, you are welcome to contact the researchers to discuss it further. In the unlikely event if you find the personality test distressing, you can ask to be referred for counselling. If you wish to be informed about the research outcome you can optionally check the "Wish to get a copy" checkbox at the end of the study. We will send a technical report on the research project as soon as it is available.

Consent

To participate, you must check the "I give consent" checkbox below.

☐ I give consent

Appdx B

Survey Questionnaire

A survey of key factors affecting effectiveness of software testing professionals

PERSONAL INFORMATION

1.1 Sex: (Please check one option) ☐ Female ☐ Male

1.2 Age:

1.3 Nationality:

1.4 Education: (Please choose one option from below)

- ☐ University Degree in Software Engineering
- ☐ University Degree in Computer Science
- ☐ University Degree in Other IT fields
- ☐ Associated Degree/Diploma in Software Engineering
- ☐ Associated Degree/Diploma in Computer Science
- ☐ Associated Degree/Diploma in Other IT fields
- ☐ Other Degree/Diploma
- ☐ No Degree/Diploma

2. *EMPLOYMENT INFORMATION*

2.1 Nature of Your Present Employment: (Please choose one option from below)

- ☐ Self Employed
- ☐ Employed in a Small[1] IT Company[3]
- ☐ Employed in a Large[2] IT Company [3]
- ☐ Employed in a Small[1] Non-IT Company[4]
- ☐ Employed in a Large[2] Non-IT Company[4]
- ☐ Not Employed

2.2 Your Present Job Responsibilities Include: (Please choose option(s) from below, you can choose multiple option)

- ☐ Developing software module/program on software specification and testing self developed modules/programs
- ☐ Developing software module/program on software specification and testing modules/programs developed by others
- ☐ Testing software modules/programs developed by others
- ☐ Manage Software Testers within a project
- ☐ Others , Please mention your responsibilities

2.3 Your Experience in Software Testing (Working as software tester/QA engineer): (Please choose one option from below)

☐ No experience
☐ Less than 1 year experience
☐ Between 1 and 3 years experience
☐ Between 3 and 5 years experience
☐ More than 5 years experience

[1] Small company implies having employee less than 50
[2] Large company implies having employee more than 50
[3] Company's main focus is building or maintaining software
[4] Company's main focus is not building or maintaining software

3 PERFORMANCE IN SOFTWARE TESTING

3.1 "The performance of software testers varies greatly from tester to tester!" – Do you agree with this statement? (Please choose one option from below)

☐ Completely disagree
☐ Somewhat disagree
☐ Neither disagree nor agree
☐ Somewhat agree
☐ Completely agree

3.2 Please mention how much do you agree with the following factors about their importance in measuring the performance of software testers. (Please choose one option for each factor)

Factors Important in Measuring Performance of Software Testers						
a	Number of bugs found	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
b	Severity of bugs	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
c	Quality of bug report	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
d	Ability of bug advocacy	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
e	Rigorousness of test planning and execution	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree

3.2.1 If you think there is any other factor(s) rather than specified in the above list, please mention them:

3.3 Please mention how much do you agree with the following qualities about their influence on the performance of software testers. (Please choose one option for each factor)

Qualities Influencing Performance of Software Testers						
a	Knowledge of specific testing techniques	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
b	Expertise in the problem domain	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
c	Testing specific training/certification	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
d	Intelligence	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
e	Dedication	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
f	Punctuality /Time value	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
g	Thoroughness	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
h	Positive Attitude	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
i	Interpersonal skills	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree

3.3.1 If you think there is any other important quality(s) not specified in the above list, please list them:

3.4 Do you think good programming skills help to improve performance as a tester?

☐ Absolutely
☐ Somewhat
☐ May be or not
☐ Not much
☐ Never

3.5 Do you think academic record is a good predictor of performance of software testers? (Please choose one option from below)

☐ Absolutely
☐ Somewhat
☐ May be or not
☐ Not much
☐ Never

3.6 Consider the best software tester you have worked with. How would you compare the value of their contribution to a project to that of an “average” tester? Are they approximately (Please choose one option from below)

a) 20% more efficient to the project
b) 50% more efficient to the project
c) 80% more efficient o the project
d) 100% more efficient to the project
e) Some other percentage, please specify

3.7 In your opinion, what can help to improve your performance as a tester?

3.8 What your employer can do to improve your performance in your role of software testing?

4 AUTOMATION OF SOFTWARE TESTING

4.1 Do you frequently use automatic tools for software testing? (Please choose one option from below)

☐ Yes (If yes, Please check 4.1.1)
☐ No (If no, please go to 4.2)

4.1.1a Which automated tools have you used? (optional):

5 4.1.1b Which of those were most useful and why? (optional): 4.2 According to you, in what ways tools are (or are not) helpful to increase your effectiveness/productivity/performance?

EXPERIENCE IN SOFTWARE TESTING

5.1 Do you think experience in software testing is helpful to increase performance of a software tester? (Please choose one option from below)

☐ Yes
☐ No (if, no please go to next section)

5.2 Performance of software testing grows with time. For instance, “Performance of a software tester is better in the sixth month in comparison to her performance in the third month” – Do you agree? (Please choose one option from below)

☐ Completely disagree
☐ Somewhat disagree
☐ Neither disagree nor agree
☐ Somewhat agree
☐ Completely agree

5.3 Do you believe that after certain level of experience performance of a software tester is saturated that is growth of experience is not so significant anymore? (Please choose one option from below)

☐ Completely true (Please go to question 5.3.1)
☐ Somewhat true (Please go to question 5.3.1)
☐ Neither true nor false
☐ Somewhat false
☐ Completely false

5.3.1 When is this “saturation point” reached?

5.4 Please comment on the importance of experience in software testing (optional):

6 CHARACTERISTICS OF SOFTWARE TESTERS

6.1 Consider the good software testers you have worked with (or yourself, if you believe you are a good tester). Would you agree or disagree that these people have the following characteristics/nature: (Note: For the purposes of this question, we are not interested in whether these characteristics are the reason behind their good performance, just whether they have these characteristics).

(Please choose one option for each characteristic)

Characteristics/Nature of Good Software Testers						
a	Good interaction with outward social world	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
b	Openness to new experiences / Intellectual curiosity	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
c	Tendency towards negative emotionality[1]	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
d	Qualities like Trust, modesty, tender mindedness and so on	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
e	Personal organization[2]	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree

[1] By negative emotionality we mean sadness, anxiety, temper and depression

[2] By personal organization we mean discipline, competence, ambition and dutifulness.

6.2.1 If you think there are any other characteristics that are important please specify them:

7 **TRAINING*/CERTIFICATION ON SOFTWARE TESTING***

7.1 Have you, in the last five years, done any training/certification on techniques/tools of software testing? (Please choose one option from below)

☐ Yes (If yes, please go to 7.1.1)
☐ No (If yes, please go to next section)

7.1.1 What training/certification did you do? (optional)

7.2 How useful in your professional activities were the techniques/tools for you? (Please choose one option from below)

- ☐ Not at all useful
- ☐ Somewhat useful
- ☐ Neither useful or not useful
- ☐ Not very useful
- ☐ Complete waste of time

7.3 Why (or why not) were these techniques/tools useful (or not useful)?

8 SOFTWARE TESTING TEAM PERFORMANCE

8.1 Please rank (from 1 to 7) the following factors in order of their importance on recruiting members to build a software testing team. (lower score implies higher rank)

No.	Factors	Rank
1	Testing performance	
2	Interpersonal Skills	
3	Team playing capability	
4	Experience in software Testing	
5	Training/certification in software testing	
6	Knowledge of the specific problem domain	
7	Compatibility with specific other proposed team members (if known)	

8.1.1 Please mention if there is any other important factors for test team building that are not listed above?

8.2 Do you think all members of a test team should be very good testers for the team to perform well?

- ☐ Completely disagree
- ☐ Somewhat disagree
- ☐ Neither disagree nor agree
- ☐ Somewhat agree
- ☐ Completely agree

8.3 Do you think all members of a test team should be good team players for the team to perform well?

- ☐ Completely disagree
- ☐ Somewhat disagree
- ☐ Neither disagree nor agree
- ☐ Somewhat agree
- ☐ Completely agree

8.4 Do you think a diverse test team helps to improve performance?

☐ Yes (if yes please check 8.5)

☐ No (if no please go to the next section)

8.5 In order to build a good software test team, which kind of diversity would you look for?

Factors for selecting members to build a Software Testing team						
a	Diversity of personality	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
b	Diversity of professional background/experience	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
c	Diversity of Age	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree
d	Diversity of communicational skills	☐ Completely disagree	☐ Somewhat disagree	☐ Neither disagree nor agree	☐ Somewhat agree	☐ Completely agree

8.5.1 If there are other diversities that you would look for building a test team, please state them:

8.6 Do you think a test team performs better when they have experience working as a team, rather than gathering experience as individuals?

☐ Completely disagree
☐ Somewhat disagree
☐ Neither disagree nor agree
☐ Somewhat agree
☐ Completely agree

Optional Question:

Do you think there are any other key factors influencing the performance of software testers, that are not covered by our questionnaire?

In your opinion, what makes a good software tester?
If you wish to be informed of the results of our survey, you can provide your email address and we will send you a copy of our findings. The address will be stored separately and NOT associated with your responses to questions.

Email Address (optional):

Survey Questionnaire

Personality Traits of Software Testers

Demographic information of the participant:

Please provide the following information about you:

Question	**Type**	**Options to choose from**
Gender	closed	• Male • Female
Age range	closed	• 25-30 • 31-40 • 41-50 • 51-60 • 61+
Country of work (Where do you work?)	closed	List of countries
Main job responsibility	closed	• Developing modules/programs based on software specification • Testing modules/programs developed by others • Managing developers • Managing testers • Other, please specify ....
Duration of experience in software testing/development	closed	• Less than a year • From 1 to 3 years • From 3-5 years • More than 5 years
Nature of employment	closed	• Self employed • Employed in small IT company • Employed in large IT company • Employed in small non IT company • Employed in large non IT company

Personality Assessment

No	items	Very inaccurate	Inaccurate	Neither inaccurate nor accurate	Accurate	Very accurate
1	Often feel blue.					
2	Dislike myself.					
3	Am often down in the dumps.					
4	Have frequent mood swings.					
5	Panic easily.					
6	Rarely get irritated.					
7	Seldom feel blue.					
8	Feel comfortable with myself.					
9	Am not easily bothered by things.					
10	Am very pleased with myself.					
11	Feel comfortable around people.					
12	Make friends easily.					
13	Am skilled in handling social situations.					
14	Am the life of the party.					
15	Know how to captivate people.					
16	Have little to say.					
17	Keep in the background.					

18	Would describe my experiences as somewhat dull.					
19	Don't like to draw attention to myself.					
20	Don't talk a lot.					
21	Believe in the importance of art.					
22	Have a vivid imagination.					
23	Tend to vote for liberal political candidates.					
24	Carry the conversation to a higher level.					
25	Enjoy hearing new ideas.					
26	Am not interested in abstract ideas.					
27	Do not like art.					
28	Avoid philosophical discussions.					
29	Do not enjoy going to art museums.					
30	Tend to vote for conservative political candidates.					
31	Have a good word for everyone.					
32	Believe that others have good intentions.					
33	Respect others.					
34	Accept people as they are.					
35	Make people feel at ease.					
36	Have a sharp tongue.					
37	Cut others to pieces.					

38	Suspect hidden motives in others.					
39	Get back at others.					
40	Insult people.					
41	Am always prepared.					
42	Pay attention to details.					
43	Get chores done right away.					
44	Carry out my plans.					
45	Make plans and stick to them.					
46	Waste my time.					
47	Find it difficult to get down to work.					
48	Do just enough work to get by.					
49	Don't see things through.					
50	Shirk my duties.					

Survey Questionnaire

Performance Appraisal of Software Testers

Demographic information of the participant:

Please provide the following information about you:

Question	Type	Options to choose from
Gender	closed	• Male • Female
Age range	closed	• 25-30 • 31-40 • 41-50 • 51-60 • 61+
Country of Work (Where do you work?)	closed	List of countries
Duration of experience in managing testers	closed	• Less than a year • From 1 to 3 years • From 3-5 years • More than 5 years
Nature of employment	closed	• Self employed • Employed in small IT company • Employed in large IT company • Employed in small non IT company • Employed in large non IT company

Information about the organization:

Please provide the following information about the organization you work in:

Question	Type	Options to choose from
Number of employee	Open	
Business domain	Closed	• Web application development • Game development • Smart phone application development

		• Desktop application development • Other (with an option to provide details)
Development process	Closed	• Waterfall • Prototyping • Agile • XP • Scrum • Other (with an option to provide details)
Does the organization follow any process improvement standard?	Closed	• No • Yes (with an option to provide details)

Survey questionnaire (Practice of performance appraisal):

Please provide your response to the following:

Number	Question	Type	Options to choose from
1	Does the organization you work in practice formal process of employee appraisal? Formal process is structured and periodic review of efficiency of carrying out the duties assigned to the employees.	closed	• Yes • No
2	Does the organization you work in use specialized performance appraisal method/form for software testers?	closed	• Yes • No
2.1	If no, how do you appraise the performance of software testers?	open	N/A
2.2	If yes, please describe the appraisal method the organization use for software testers.	open	N/A
2.3	If yes, is the appraisal method for software testers developed in-house by the organization you work in?	closed	• Yes • No

2.3.1	If the appraisal method/form is not developed in-house by the organization, where does the organization render the method/form from?	Open	N/A
2.4	If yes, do you think the method/form can appraise the performance of software testers appropriately?	closed	• Yes • No
2.5	If yes, do you think the performance components taken in to account in the method/form are sufficient?	closed	• Yes • No

Survey questionnaire (Opinion on performance appraisal of testers):

1	What performance components do you think should be taken in to account for performance appraisal of testers?	Open	N/A
2	What are the advantages of taking your proposed components in to account?	Open	N/A
3	What can be the disadvantages of taking your proposed components in to account?	Open	N/A

Question on follow-up study:

Are you interested to participate in a follow-up study of performance appraisal of testers?

- Yes
- No

Appdx C

SUHREC Project 2009/273 Ethics Clearance

To: Dr R Merkel, FICT / Ms Tanjila Kanij

Dear Dr Merkel,

SUHREC Project 2009/273 A survey of key factors affecting effectiveness of software testing professionals
Approved Duration: 18/12/2009 To 18/12/2013 [Adjusted]

I refer to the ethical review of the above project protocol undertaken on behalf of Swinburne's Human Research Ethics Committee (SUHREC) by SUHREC Subcommittee (SHESC2) at a meeting held on 27 November 2009. Your responses to the review as e-mailed on 8 December were forwarded to a SHESC2 delegate for review.

I am pleased to advise that, as submitted to date, the project has approval to proceed in line with standard on-going ethics clearance conditions here outlined.

- All human research activity undertaken under Swinburne auspices must conform to Swinburne and external regulatory standards, including the National Statement on Ethical Conduct in Human Research and with respect to secure data use, retention and disposal.

- The named Swinburne Chief Investigator/Supervisor remains responsible for any personnel appointed to or associated with the project being made aware of ethics clearance conditions, including research and consent procedures or instruments approved. Any change in chief investigator/supervisor requires timely notification and SUHREC endorsement.

- The above project has been approved as submitted for ethical review by or on behalf of SUHREC. Amendments to approved procedures or instruments ordinarily require prior ethical appraisal/ clearance. SUHREC must be notified immediately or as soon as possible thereafter of (a) any serious or unexpected adverse effects on participants and any redress measures; (b) proposed changes in protocols; and (c) unforeseen events which might affect continued ethical acceptability of the project.

- At a minimum, an annual report on the progress of the project is required as well as at the conclusion (or abandonment) of the project.

- A duly authorised external or internal audit of the project may be undertaken at any time.

Please contact me if you have any queries about on-going ethics clearance. The SUHREC project number should be quoted in communication. Chief Investigators/Supervisors and Student Researchers should retain a copy of this e-mail as part of project record-keeping.

Best wishes for the project.

Kaye Goldenberg
Secretary, SHESC2
****

Kaye Goldenberg
Administrative Officer (Research Ethics)
Swinburne Research (H68)
Swinburne University of Technology
P O Box 218
HAWTHORN VIC 3122
Tel +61 3 9214 8468
Fax +61 3 9214 5267

SUHREC Project 2011/036 Ethics Clearance

To: Prof. John Grundy, FICT/Ms Tanjila Kanij

Dear Prof. Grundy,

SUHREC Project 2011/036 What responsibilities software testing includes and who is better doing what?
Prof. John Grundy, FICT/Ms Tanjila Kanij
Approved Duration: 27/06/2011 To 31/09/2013 [Adjusted]

I refer to the ethical review of the above project protocol undertaken on behalf of Swinburne's Human Research Ethics Committee (SUHREC) by SUHREC Subcommittee (SHESC1) at a meeting held on 25 March 2011. Your responses to the review as e-mailed on 10 May and 6, 21 and 23 June 2011 were put to a nominated SHESC1 delegate for review.

I am pleased to advise that, as submitted to date, the project has approval to proceed in line with standard on-going ethics clearance conditions here outlined.

- All human research activity undertaken under Swinburne auspices must conform to Swinburne and external regulatory standards, including the National Statement on Ethical Conduct in Human Research and with respect to secure data use, retention and disposal.

- The named Swinburne Chief Investigator/Supervisor remains responsible for any personnel appointed to or associated with the project being made aware of ethics clearance conditions, including research and consent procedures or instruments approved. Any change in chief investigator/supervisor requires timely notification and SUHREC endorsement.

- The above project has been approved as submitted for ethical review by or on behalf of SUHREC. Amendments to approved procedures or instruments ordinarily require prior ethical appraisal/ clearance. SUHREC must be notified immediately or as soon as possible thereafter of (a) any serious or unexpected adverse effects on participants and any redress measures; (b) proposed changes in protocols; and (c) unforeseen events which might affect continued ethical acceptability of the project.

- At a minimum, an annual report on the progress of the project is required as well as at the conclusion (or abandonment) of the project.

- A duly authorised external or internal audit of the project may be undertaken at any time.

Please contact me if you have any queries about on-going ethics clearance. The SUHREC project number should be quoted in communication. Chief Investigators/Supervisors and Student Researchers should retain a copy of this e-mail as part of project record-keeping.

Best wishes for the project.

Yours sincerely

Kaye Goldenberg
Secretary, SHESC1
****

Kaye Goldenberg
Administrative Officer (Research Ethics)
Swinburne Research (H68)
Swinburne University of Technology
P O Box 218
HAWTHORN VIC 3122
Tel +61 3 9214 8468

SUHREC Project 2010/144 Ethics Clearance

To: Dr Robert Merkel, FICT/ Miss Tanjila Kanij
CC: Ms Mandish Webb, FICT

Dear Dr Merkel,

SUHREC Project 2010/144 The Effect of Personality Traits on Effectiveness of Software Testing
Dr Robert Merkel, FICT/ Miss Tanjila Kanij
Approved Duration: 10/08/2010 To 10/08/2013 [Adjusted]

I refer to the ethical review of the above project protocol undertaken on behalf of Swinburne's Human Research Ethics Committee (SUHREC) by SUHREC Subcommittee (SHESC2) at a meeting held on 19 July 2010. Your response to the review as e-mailed on 3 August 2010 was put to a nominated SHESC2 delegate for consideration. *The Delegate wished me to convey to you their compliments on an excellent response.*

I am pleased to advise that, as submitted to date, the project has approval to proceed in line with standard on-going ethics clearance conditions here outlined.

- All human research activity undertaken under Swinburne auspices must conform to Swinburne and external regulatory standards, including the *National Statement on Ethical Conduct in Human Research* and with respect to secure data use, retention and disposal.

- The named Swinburne Chief Investigator/Supervisor remains responsible for any personnel appointed to or associated with the project being made aware of ethics clearance conditions, including research and consent procedures or instruments approved. Any change in chief investigator/supervisor requires timely notification and SUHREC endorsement.

- The above project has been approved as submitted for ethical review by or on behalf of SUHREC. Amendments to approved procedures or instruments ordinarily require prior ethical appraisal/ clearance. SUHREC must be notified immediately or as soon as possible thereafter of (a) any serious or unexpected adverse effects on participants and any redress measures; (b) proposed changes in protocols; and (c) unforeseen events which might affect continued ethical acceptability of the project.

- At a minimum, an annual report on the progress of the project is required as well as at the conclusion (or abandonment) of the project.

- A duly authorised external or internal audit of the project may be undertaken at any time.

Please contact me if you have any queries about on-going ethics clearance. The SUHREC project number should be quoted in communication. Chief Investigators/Supervisors and Student Researchers should retain a copy of this e-mail as part of project record-keeping.

Best wishes for the project.

Yours sincerely

Kaye Goldenberg
Secretary, SHESC2
****

SUHREC Project 2011/172 Ethics Clearance

To: Prof J Grundy Ms Tanjila Kanij FICT
CC: Ms Mandish Webb Research Administration Coordinator FICT

Dear John and Tanjila,

SUHREC Project 2011/172 Performance appraisal of software tester
Prof J Grundy Ms Tanjila Kanij FICT
Approved duration: 4/11/2011 To 4/11/2013 [Adjusted]

I refer to the ethical review of the above project protocol undertaken by a SUHREC Subcommittee (SHESC3). Your responses to the reviews, as e-mailed on 5 September and 31 October 2011, were put to and approved by SUHREC delegate(s).

I am pleased to advise that, as submitted to date, the project may proceed in line with standard on-going ethics clearance conditions here outlined.

- All human research activity undertaken under Swinburne auspices must conform to Swinburne and external regulatory standards, including the current National Statement on Ethical Conduct in Research Involving Humans and with respect to secure data use, retention and disposal.

- The named Swinburne Chief Investigator/Supervisor remains responsible for any personnel appointed to or associated with the project being made aware of ethics clearance conditions, including research and consent procedures or instruments approved. Any change in chief investigator/supervisor requires timely notification and SUHREC endorsement.

- The above project has been approved as submitted for ethical review by or on behalf of SUHREC. Amendments to approved procedures or instruments ordinarily require prior ethical appraisal/ clearance. SUHREC must be notified immediately or as soon as possible thereafter of (a) any serious or unexpected adverse effects on participants and any redress measures; (b) proposed changes in protocols; and (c) unforeseen events which might affect continued ethical acceptability of the project.

- At a minimum, an annual report on the progress of the project is required as well as at the conclusion (or abandonment) of the project.

- A duly authorised external or internal audit of the project may be undertaken at any time.

Please contact me if you have any queries about on-going ethics clearance. The SUHREC project number should be quoted in communication. Chief Investigators/Supervisors and Student Researchers should retain a copy of this email as part of project record-keeping.

Best wishes for project.

Yours sincerely,

Ann Gaeth
Secretary, SHESC3

Dr Ann Gaeth
Administrative Officer (Research Ethics)
Swinburne Research (H68)
Swinburne University of Technology
P.O. Box 218
HAWTHORN VIC 3122
Tel: +61 3 9214 5935
Fax: +61 3 9214 5267

SUHREC Project 2012/243 Ethics Clearance

To: Professor John Grundy; FICT
Ms Tanjila Kanij

Dear John and Tanjila,

SUHREC Project 2012/243 Personality traits of software testers
Prof John Grundy, Ms Tanjila Kanij, Dr Robert Merkel
Approved Duration: 02/11/2012 To 31/01/2014 [Adjusted]

I refer to the ethical review of the above project protocol undertaken by a SUHREC Subcommittee (SHESC3). Your responses to the review, as emailed on 29 October 2012 with attachments including revised consent instrument, were put to a SHESC3 delegate for consideration.

I am pleased to advise that, as submitted to date, the project may proceed in line with standard on-going ethics clearance conditions here outlined.

- All human research activity undertaken under Swinburne auspices must conform to Swinburne and external regulatory standards, including the current *National Statement on Ethical Conduct in Human Research* and with respect to secure data use, retention and disposal.

- The named Swinburne Chief Investigator/Supervisor remains responsible for any personnel appointed to or associated with the project being made aware of ethics clearance conditions, including research and consent procedures or instruments approved. Any change in chief investigator/supervisor requires timely notification and SUHREC endorsement.

- The above project has been approved as submitted for ethical review by or on behalf of SUHREC. Amendments to approved procedures or instruments ordinarily require prior ethical appraisal/ clearance. SUHREC must be notified immediately or as soon as possible thereafter of (a) any serious or unexpected adverse effects on participants and any redress measures; (b) proposed changes in protocols; and (c) unforeseen events which might affect continued ethical acceptability of the project.

- At a minimum, an annual report on the progress of the project is required as well as at the conclusion (or abandonment) of the project.

- A duly authorised external or internal audit of the project may be undertaken at any time.

Please contact the Research Ethics Office if you have any queries about on-going ethics clearance. The SUHREC project number should be quoted in communication. Chief Investigators/Supervisors and Student Researchers should retain a copy of this email as part of project record-keeping.

Best wishes for project.

Yours sincerely,

Sheila Hamilton-Brown
Secretary, SHESC3

****
Sheila Hamilton-Brown
Administrative Officer (Research Ethics & Biosafety)
(Tues, Wed & Fri)
Swinburne Research (H68)
Swinburne University of Technology
PO Box 218
HAWTHORN VIC 3122
Tel: 03 9214 5935
Fax: 03 9214 5267