2014 Australasian Conference on Software Engineering (ASWEC 2014), Sydney, Australia, April 2014, © IEEE CS Press.

A Preliminary Survey of Factors Affecting Software Testers

Tanjila Kanij
Swinburne University of Technology
Hawthorn, Victoria
Australia
Email:tkanij@swin.edu.au

Robert Merkel
Monash University
Clayton, Victoria
Australia
Email: robert.merkel@monash.edu

John Grundy
Swinburne University of Technology
Hawthorn, Victoria
Australia
Email:jgrundy@swin.edu.au

Abstract—**Most software testing research has focused on the development of systematic, standardised, and automated testing methodologies and tools. The abilities and expertise needed to apply such techniques and tools - such as personality traits, education, and experience - have attracted a comparatively small amount of research attention. However, the limited research in the area to date provides some indication that the human traits of software testers are important for effective testing. This paper presents the opinions of software testers themselves, collected through an online survey, on the importance of a variety of factors that influence effective testing, including testing-specific training, experience, skills, and human qualities like dedication and general intelligence. The survey responses strongly suggest that while testing tools and training are important, human factors were similarly considered highly important. Domain knowledge, experience, intelligence, and dedication, amongst other traits, were considered crucial for a software tester to be effective. As such, while systematic methodologies are important, the individual most clearly does matter in software testing. The results of our research have implications for education, recruitment, training and management of software testers.**

I. Introduction

Software testing is a crucial part of the process of producing high quality, reliable software systems. It can consume more than fifty percent of the total development effort [1]. Despite decades of research in the field, effective and economic software testing remains a challenge. 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 [2]. 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 and to interpret testing process outcomes.

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 [3], 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 software tester by increasing the amount of testing achievable. According to Armour [4], good software testers have a *nose* for testing, possessing "a kind of intuition that tells them what to test and how". Such ideas find full expression in the concept of exploratory testing [5], unstructured testing described as "simultaneous learning, test design and test execution". Beer and Ramler [6] studied the importance of experience in software testing. They found that experienced software testers have higher level of domain knowledge that helps support effective testing. According to their findings experience helps a software tester more accurately interpret the specification and to make better use of customised tools. However, McDaniel, Schmidt, and Hunter [7], in a general survey across a broad range of professions of the connection between experience and job performance, noted a correlation between experience and job performance. They further observed that the correlation became weaker as the length of experience increased, giving rise to the possibility that beyond a certain point, additional experience may not make much difference. da Cunha and Greathead [8] reported a connection between personality, as measured by psychological testing, and debugging performance. Overall, there is considerable support in the literature for an alternative view of software testing as a task critically dependent on the personal characteristics of the software tester.

The relative importance of these factors is of crucial importance both for researchers and practitioners. For practising software testers and those who manage testing teams, if human factors are key, identifying, motivating, and rewarding good software testers is the crucial determining factor in a successful testing program. If techniques and tools are most significant, the selection and reward of individuals is less important compared to appropriate training in the best techniques, and the use of the most appropriate automated tools that support these techniques. For researchers, if human factors are more important than currently appreciated, a new research program opens up: identifying the most important such factors, and developing ways to identify individuals who have such traits if they are innate, and how to develop these traits if they are indeed teachable.

In reconciling these two viewpoints - techniques and tools on one hand, human factors on the other - we believe that the views of practising software testers are a good place to start. This paper presents our attempt to assess these views: a survey of the opinion of software testing practitioners on the factors influencing their effectiveness. The questionnaire included sections of questions on many of possible factors influencing effectiveness of software testers, as well as open-ended questions where survey respondents could mention other factors not explicitly listed. Software testing professionals from around the world participated voluntarily in the survey and shared their perceptions with us.

The paper is organised as follows: Section II gives a brief description of the survey and Section III presents the results in detail. Section IV analyses some possible threats to the validity of the survey and Section V discusses the results. Finally, Section VI concludes the paper including directions for future research.

II. Study Design

While there are a number of different techniques for soliciting information in social research contexts, such as this, a survey was considered appropriate for this initial study. This is because it allows the views of a wide range of people to be collected in a feasible, timely manner [9].

A. Sample Selection

Our goal was to survey the perspectives of software testing professionals. However, it is not only software testers themselves who may have an opinion on effective software testing; managers of software testers, software developers who interact with software testers, or developers who test their own software may also have valuable insights. As noted in the introduction, our primary goal was to seek the opinion of software testers themselves.

To select the sample for the survey we have searched for software testing related LinkedIn and Yahoo! groups. The groups matching the keyword "software testing" were listed using an automated script. From the description of the listed groups, 21 Yahoo! and 29 LinkedIn groups were selected using purposive sampling [9]. In this sampling process we examined the description of each of the groups and selected those that were active and were solely software testing related, and excluded groups specifically relating to employment. After seeking permission from group moderators and receiving permission from 12 Yahoo! And 12 LinkedIn groups, we sent invitations to participate to group members, providing a link to our online survey. No financial or other reward was offered for participation in the survey.

B. Questionnaire Design

The survey questionnaire was designed to elicit the respondents' opinions on some of the factors that our intuitions and literature review suggested might affect the effectiveness of a software tester. To elicit these responses, the survey included both closed and open questions [9]. While closed-form questions are easier to summarize, open questions allowed respondents to point out issues that were not mentioned in the closed-form questions. Most closed survey questions used a Likert scale [10] with five possible responses from "completely disagree" to "completely agree".

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

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

2) Employment Information: The questions in this section asked respondents about their type of present employment, main job responsibilities and amount of experience they have in the present role. We also asked the respondents about the size of their employer (a "large" company defined as having 50 or more employees, whereas a small one had fewer than 50), and whether they worked for a software company (defined as a company whose main focus was building or maintaining software).

3) Performance of Software Tester: 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 addressed whether programming skill and academic track record are connected with software testing effectiveness.

4) Automation of Testing: This section asked the respondents about their use of automated testing tools. Questions included how frequently they used such tools, which tools they used, and in what ways the tools helped to improve their effectiveness.

5) Experience in Software Testing: This section asked respondents about their view of experience in software testing. The questions asked about the importance of experience, and whether the benefits of experience accumulated only until some point of "saturation", after which no additional benefit was gained.

6) Characteristics of Software Testers: This section asked respondents' opinions on the personality characteristics of good software testers. The characteristics listed were based on the well-known "Big Five" factor model of personality, which is one of the most popular models of personality traits in modern personality psychology research [11]. This model groups the many traits that can be used to describe an individual's personality into five broad dimensions, *Extraversion*, *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 (using a Likert scale) whether they believed that the individual considered exhibited this characteristic.

7) Training/Certification in Software testing: To address the importance of training and certification, we asked respondents whether they had done any training or certification related to software testing in the last five years and how useful the training/certification was to them.

8) Software Test Team Building: This section asked questions about testing as a team activity, including a general ranking of important factors for individuals to possess when working as part of a testing team, whether various types

TABLE I. AGE RANGES (QUESTION 1.2)

18-30 years	39.8%
31-40 years	35%
41-50 years	17.5%
51-65 years	2.9%
No Response	4.9%

of diversity within the test team is desirable, and whether experience working together as a team is important.

For space reasons, the results of this section are not included in this article and have been published in a separate article [12].

C. Pilot study and ethics approval

A pilot project on a selected sample software engineers was conducted. In the pilot project the questionnaire was sent to 7 software engineers requesting to comment on the questionnaire. 5 of them gave us their opinion about the questionnaire. From their feedback, changes including adding one survey subsection, and additional questions, as well as rewording of several other questions, were made.

At Swinburne University of Technology, any research activity conducted with human needs to undergo ethical review. The final version of the questionnaire was approved on behalf of Swinburne's Human Research Ethics Committee (SUHREC) by a delegated SUHREC subcommittee (SHESC2).

D. Implementation

The survey itself was conducted online, with the invitation emails containing a link to the survey site.

The online site and the data analysis tool were implemented using custom-built PHP, JavaScript and HTML scripts. The responses are stored in a MySQL database. The survey is available at [13]. We chose to use a custom data collection and analysis tool to allow us to very flexibly present and analyse data from the survey. It also allowed us to ensure a level of data protection as specified in our Ethics clearance application.

III. RESULTS

The data was collected during July 2010 to March 2011. A total of 104 respondents completed the survey. As the respondents had the option to leave questions blank, our results report how many respondents did not answer each question.

A. Personal Information

The majority of the respondents (71.8%) were male, with 23.3% female participants, and 4.9% of participants who did not indicate their gender. More than half of the respondents were between 18-40 years of age. Table III shows 28.8% of respondents reported their "Country" as India, with the second largest group of respondents (24%) coming from the United States.

Table II indicates the educational attainment of the respondents. The vast majority had a university degree in software engineering or another IT-related field, with a substantial minority possessing a graduate degree.

TABLE II. EDUCATIONAL INFORMATION (QUESTION 1.4)

University Degree in Software Engineering	26%
University Degree in Another IT Field	14.4%
Graduate Degree in Software Engineering	16.3%
Graduate Degree in Another IT Field	10.6%
Associated Degree/Diploma in Software Engineering	4.8%
Associated Degree/Diploma in Another IT Field	3.8%
Other Degree/Diploma	16.3%
No Degree/Diploma	4.8%
No Response	2.9%

TABLE III. COUNTRY OF THE RESPONDENTS (QUESTION 1.3)

India	28.8%
United States of America	24%
Bangladesh	9.6%
Netherland	4.8%
United Kingdom, Pakistan, Finland	2.9% each
Sweden, Romania, New Zealand, Brazil, Israel	1.9% each
UAE, Egypt, Switzerland, Philippines, Mexico, Poland, Austria, Australia, Denmark, Ireland	1% each
No Response	3.8%

B. Employment Information

Table IV reports the employment status of the respondents. Nearly 60% of respondents were employed by large IT companies; with a little under 20% of the sample employed by smaller IT companies, and a similar proportion employed by larger "non-IT companies". Table V shows that almost half of the respondents (49%) had more than five years of job experience.

Table VI indicates the respondents' job responsibilities. Note that respondents were able to select more than one option. 76.9% of respondents indicated that they were responsible for 'testing software modules/programs developed by others'. 45.2% 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 software testers, and business development, amongst others.

C. Performance

The vast majority of respondents agreed that performance of software testing varies from tester to tester, as indicated in Table VII. Respondents believed that the difference was substantial, with 35% stating that the best software tester they had worked with was "50% more valuable to the project", and 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.

1) Measurement of Performance: Table IX summarizes the respondents' views on the importance of various factors that might be used for measuring the effectiveness of software testers. It appears from the responses that the majority of the respondents have at least somewhat agreed that all of the factors listed here are important to the testing role. Among the listed factors *quality of bug report* is considered important factor by almost all the respondents (92.3%). *Bug advocacy* (83.7%), *Rigorousness of test planning and execution* (78.8%)

TABLE IV. EMPLOYMENT TYPE (QUESTION 2.1)

Self Employed	1%
Employed in a Small IT Company	18.3%
Employed in a Large IT Company	58.7%
Employed in a Small Non-IT Company	0%
Employed in a Large Non-IT Company	17.3%
Not Employed	2.9%
No Response	1.9%

TABLE V. EXPERIENCE (QUESTION 2.3)

No experience	1%
Less than 1 years experience	5.8%
Between 1 and 3 years experience	18.3%
Between 3 and 5 years experience	24%
More than 5 years experience	49%
No Response	1.9%

and *Severity of bugs found* (76%) follow the list. The response on the factor "number of bugs found" was more mixed - 34% of the respondents at least somewhat disagree that *number of bugs found* is an important factor for measuring performance of software testers. 31.7% of the total respondents, in the accompanying open-ended question, indicated other factors that they think important in measuring performance of software testers. A review of the factors mentioned by the respondents shows that 6.7% mentioned domain knowledge (including the problem domain, the product, and the relevant testing techniques) as an important factor. A similar number of respondents mentioned understanding requirements and the quality of the communication with developers.

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, only one respondent opined that the performance of a software tester can be measured by the number of bugs reported in the "live" environment (after deployment). Ultimately, this is probably the key measure for software testing activity - ensuring deployed software has minimal number and criticality of errors.

2) Importance of programming skill and academic performance: A majority of respondents do not think that programming skill helps to improve performance as a software tester, with 67% disagreeing, and less than 12% agreeing, as shown in Table X.

As table XI shows, there was a somewhat mixed response on whether academic record is 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.

3) Contributors to Performance: This question asked respondents to indicate the extent to which they agreed that various human qualities might influence the performance of software testers. Table XII summarizes the responses. These show that most respondents agreed that the qualities mentioned were influential. Almost all the respondents (92.4%) agreed that *good knowledge of the problem domain* is an important quality for software testing. *Intelligence* (90.4%), *dedication* (90.4%), *Thoroughness* (88.4%), *Knowledge of specific testing technique* (86.5%), *positive attitude* (83.6%), *interpersonal skill* (83.7%) were considered influential by the vast majority

TABLE VI. MAIN JOB RESPONSIBILITIES (QUESTION 2.2)

Developing software module/program on software specification and testing self developed modules/programs	7.7%
Developing software module/program on software specification and testing modules/programs developed by others	11.5%
Testing software modules/programs developed by others	76.9%
Manage Software Testers within a project	46.2%
Others	19.2%
No Response	2.9%

TABLE VII. RESPONSES ON "PERFORMANCE VARIES A LOT FROM TESTER TO TESTER" (QUESTION 3.1)

Completely Disagree	7.7%
Somewhat Disagree	3.8%
Neither Agree Nor Disagree	3.8%
Somewhat Agree	26%
Completely Agree	56.7%
No Response	1.9%

of responses, and *punctuality* (70.2%) was also considered influential by a very clear majority of respondents. Respondents were less clear on the influence of *testing specific training/certification* - 44% agreed that it was influential, where 25% disagreed and 27% were not sure.

(13.5%) respondents mentioned some other qualities. The most common one was "motivation"; others included "Accepting new challenges", "Automation of testing", "ability to work under pressure", "knowledge sharing and good communication skill", and "out of box thinking".

a) 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, software testers must be dedicated and make an active effort to improve their work; as one respondent put it, software testers should "love testing". Many respondents mentioned the need for learning, including both new testing techniques, and about the problem and business domain of their work. Learning from one's own personal experience, as well as the experiences of other software testers, were considered important by respondents.

b) 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 (a number of respondents mentioned direct customer contact as important), 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 software testers to do well

TABLE IX. RESPONSES ON "FACTORS IMPORTANT IN MEASURING PERFORMANCE OF SOFTWARE TESTERS" (QUESTION 3.2)

	Completely Disagree	Somewhat Disagree	Neither Disagree Nor Agree	Somewhat Agree	Completely Agree	No Response
Number of Bugs Found	22.3%	11.7%	12.8%	47.9%	13.8%	2.1%
Severity of bugs	9.6%	2.9%	9.6%	35.6%	40.4%	1.9%
Quality of bug report	1%	1%	2.9%	31.7%	60.6%	2.9%
Bug Advocacy	1%	2.9%	10.6%	35.6%	48.1%	1.9%
Rigorousness of test planning and execution	0%	3.8%	14.4%	29.8%	49%	2.9%

TABLE XII. RESPONSES ON "QUALITIES INFLUENCING PERFORMANCE OF SOFTWARE TESTERS (QUESTION 3.3)

	Completely Disagree	Somewhat Disagree	Neither Disagree Nor Agree	Somewhat Agree	Completely Agree	No Response
Knowledge of specific testing techniques	1%	5.8%	3.8%	37.5%	49%	2.9%
Expertise in the problem domain	1%	1%	2.9%	33.7%	58.7%	2.9%
Testing specific training/certification	12.5%	12.5%	26.9%	31.7%	12.5%	3.8%
Intelligence	0%	0%	7.7%	35.6%	54.8%	1.9%
Dedication	0%	1%	3.8%	20.2%	70.2%	4.8%
Punctuality /Time value	4.8%	4.8%	16.3%	27.9%	41.3%	4.8%
Thoroughness	0%	0%	6.7%	24%	64.4%	4.8%
Positive Attitude	2.9%	2.9%	7.7%	23.1%	60.6%	2.9%
Interpersonal Skill	0%	1%	9.6%	38.5%	45.2%	5.8%

TABLE VIII. COMPARED TO AN "AVERAGE" SOFTWARE TESTER, THE BEST SOFTWARE TESTER YOU HAVE WORKED WITH IS... (QUESTION 3.6)

20% more valuable to the project	9.6%
50% more valuable to the project	34.6%
80% more valuable to the project	26.9%
100% more valuable to the project	13.5%
x(!)% more valuable to the project	12.5%
No Response	2.9%

TABLE X. RESPONSES TO QUESTION "DO YOU THINK GOOD PROGRAMMING SKILLS HELP TO IMPROVE PERFORMANCE AS A SOFTWARE TESTER?" (QUESTION 3.4)

Completely Disagree	30.8%
Somewhat Disagree	36.5%
Neither Agree Nor Disagree	18.3%
Somewhat Agree	8.7%
Completely Agree	2.9%
No Response	2.9%

TABLE XI. RESPONSES ON "ACADEMIC RECORD IS A GOOD PREDICTOR" (QUESTION 3.5)

Completely Disagree	3.8%
Somewhat Disagree	18.3%
Neither Agree Nor Disagree	24%
Somewhat Agree	36.5%
Completely Agree	14.4%
No Response	2.9%

in their job, and provide adequate recognition of good work.

D. Influence of Automated Tools

In response to Question 4.1, 62.1% of the respondents of 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*. QuickTest Professional (QTP) is a functional and regression testing tool that provides a graphical interface to create, execute, and report the results of test scripts. *QTP* can be customised by manually writing scripts to drive testing. Selenium is a web testing framework that also provides a GUI to generate scripts for web testing, with an interface to run web testing scripts in a variety of scripting languages. 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 (31.7%) 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, 13.6% of respondents mentioned that automated tools can help doing "repetitive and mundane tests" - presumably, this relates both to time savings and to reducing boredom. 6.7% 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 - 11.5% 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 5.8% think these are helpful for load and performance testing. 5.8% respondents opined 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 (4.8%) indicated reservations about automated tools. Most of the respondents of this group opined that automated tools often require excessive (and thus costly) maintenance. They also said that some automated tools sometimes produce large scripts. One 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

TABLE XIII. RESPONSES ON "PERFORMANCE GROWS WITH EXPERIENCE" (QUESTION 5.1)

Completely Disagree	1%
Somewhat Disagree	16.3%
Neither Agree Nor Disagree	14.4%
Somewhat Agree	41.3%
Completely Agree	25%
No Response	1.9%

TABLE XIV. RESPONSES ON "PERFORMANCE IS SATURATED AFTER SOME EXPERIENCE" (QUESTION 5.2)

Completely true	1%
Somewhat true	20.2%
Neither Agree Nor false	15.4%
Somewhat false	22.1%
Completely false	22.1%
No Response	19.2%

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 software testers need to understand the tool very well before using it and should be able to judge when it is necessary to use automated tools. One respondent also said that tools should be simple, quick and directly related to business value. A few responses suggested that tester should write their own tools after understanding the problem domain.

E. *Experience*

As Table XIII 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 the respondents to comment on the importance of experience in software testing (Question 5.3). 39.4% of the respondents 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 software testers become great over time; terrible software testers stay terrible!".

Some respondents nominated specific reasons for the importance of experience. According to some of these, an experienced software 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 software tester, presumably assisting in the planning and execution of future testing. According to the respondents, experience helps software 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 opined that experience helps to grow adaptability in different situations. However, one thing most of the respondents emphasised is that experience is only fruitful if software testers learn from the past, including their mistakes. It was also mentioned that a variety of experience, including new challenges was important.

1) Saturation of Experience: We asked whether the respondents think performance is "saturated" after some level of experience - that is, at some point, is there no benefit to additional experience. Table XIV 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). Then 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 software 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 put it, "after retirement".

F. *Personality Characteristics of Software Testers*

Question 6.1 asked respondents to consider a good software tester they have worked with (or themselves, if they believe that they are a good tester), and to state whether they agreed that characteristics associated with the factors of the "Big Five" model of personality were exhibited by that person. The responses indicate that 70.2% respondents completely agree that good software testers should be open-minded. Respondents tend to agree with all of the five characteristics listed except "tendency to negative emotionality", were associated with the good software tester they considered. Interestingly 23% respondents at least somewhat agree that this characteristic was exhibited by the individual they considered, despite the fact that negative emotionality might be seen as undesirable. In the accompanying open-ended question, some other characteristics were also reported by the respondents. Most of them said "attention to the details" should be a characteristic of a good software tester. Some other characteristics listed by the respondents include "innate investigation traits", "skepticism", "tenacity", "loyalty", "creativity" and so on.

G. *Training/Certification*

56.7% of the 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 [14] certification, representing 32.7% 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.

1) Why (or why not) Training/Certification is important: 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; one respondent indicated that courses served as a

TABLE XV. RESPONSES ON "CHARACTERISTICS OF GOOD SOFTWARE TESTER" (QUESTION 6.1)

	Completely Disagree	Somewhat Disagree	Neither Disagree Nor Agree	Somewhat Agree	Completely Agree	No Response
Good interaction with outward social world	2.9%	3.8%	15.4%	45.2%	28.8%	3.8%
Open mindedness/Openness to new experiences/Intellectual curiosity	1%	1%	3.8%	19.2%	70.2%	4.8%
Tendency towards negative emotionality	28.8%	16.3%	26%	16.3%	6.7%	5.8%
Qualities like Trust, modesty and so on	1.9%	10.6%	16.3%	35.6%	30.8%	4.8%
Personal Organization	0%	1%	11.5%	33.7%	50%	3.8%

motivation to read relevant books. Some respondents thought that their training/certification helped to better understand the work, find new way and approach for testing, and helped to save testing effort. Respondents of this group advocated that the taught techniques broadened the knowledge of the testers; it depended on the tester how to apply the techniques in their specific problem domain. On the other hand, some respondents opined that the generic tools that are taught are often useless while self learned materials are considerably more useful.

IV. THREATS TO VALIDITY

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

A. Internal Validity

The pattern of responses in the characteristics and training/certification section indicates that the intended meaning and context of those questions might not have been clear enough to all respondents. 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 interpreted those questions as we had.

Another threat to 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 - if respondents did not wish to answer honestly and fully, we believe that the most likely course of action from them would have been simply to not participate in the survey. We believe that the patterns of the survey response support this view, particularly given the extended and thoughtful responses provided to the open-ended questions.

B. 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. The sample size was not particularly large for a survey questionnaire. 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. For instance, there is a possibility that younger and more sociable testers might be more likely to participate the mailing lists.

Responses were skewed towards software testers from countries where English is the main language used in technical contexts - our survey had no responses from Germany, China, or Japan, for instance. However, while we cannot be sure of the representativeness of our responses, the demographic and employment information provided suggests that for the most part respondents were experienced, professional software testers, and as such were the intended targets of our survey.

Another threat to the external validity of our research is that it seeks only the thoughts and views of expert software 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 hypotheses which can be justified with more detailed empirical studies.

V. DISCUSSION

In the opinion of our respondents, the effectiveness of individual software testers varies considerably, as discussed in Section 3.3. At least 80% of respondents believe that the "best" testers were at least 50% more valuable to the project. While it would be unwise to over-interpret this specific figure, the responses to this question alone indicate that testers themselves believe that the individual and the individual's capabilities, approach and skill matters a great deal in software testing. 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.

A number of factors that are related to the effectiveness of software testing were identified. Good domain knowledge was very strongly identified as important, and was mentioned in multiple sections by the respondents. More than 90% of respondents believed that good knowledge of the problem domain is a desired quality of a software tester. This was also explicitly mentioned by some of the respondents as important for measuring the effectiveness of software tester. Good domain knowledge is of course strongly associated with experience. This corresponds with Beer and Ramler [6]'s examination of testing practice, which showed that, in the cases studied, the program specifications provided to software testers were insufficient to construct comprehensive software test suites. Testers (or domain experts) were required to use their domain knowledge to "fill the gaps" in such incomplete specifications. As well as domain knowledge, some of our respondents expressed similar ideas to Armour's [4] "nose for testing" - in Section III-E respondents noted that experienced software testers developed the ability to identify error-prone parts of software to focus testing on. While experience was generally considered positive, merely "serving time" was not considered sufficient by our respondents. Learning from the past deeds results in continuous growth that is desired for the effectiveness of software testing. While experience is something gained from the software tester's work environment, the ability to learn from that experience seems likely to be partly innate.

In terms of recruiting and training software testers, the development of suitable expertise in the domain of application of software systems and the development of both domain expertise and testing expertise would seem to be critical. Historically testing education and training has focused on generic skills and techniques rather than using domain knowledge to e.g. design tests, interpret test results and so on. Development of expertise and experience has been assumed to happen implicitly as one practices testing. A more structured approach to gaining deep domain knowledge prior to assuming testing responsibilities for a complex system would seem to be a sensible practice. Similarly, as pointed out by some respondents, not all experience is "equal" and consideration of exposure to different aspects of testing and system under test may be useful. This may improve both tester performance and test quality, but also tester development and engagement.

From the responses to the influential qualities of software testers (Section III-C3) it is apparent that "Intelligence" and "Dedication" are considered particularly important qualities, as the significance of both of these qualities was indicated by more than 90% of the respondents. These factors are success factors in most skilled professions. *g*, the "general intelligence" commonly assessed using IQ tests, is positively correlated with job performance in a variety of job types [15], though the basis of this correlation is controversial [16]. While the extent to which intelligence is innate is also highly controversial, it is considered to have at least some innate component [17, p. 177]. "Dedication" is also interesting in that it is closely related to traits like self-discipline, responsibility, and a tendency to hard work, which are identified as facets of the *Conscientiousness (C)* trait in the "Big Five" traits from personality theory in psychology [18]. There is a very considerable body of work on the innate, social, and cultural factors affecting personality; regardless, it seems unlikely that dedication is something typically taught in a class on software testing techniques. Several other characteristics mentioned by respondents may also relate to personality traits in the Big Five model, for instance, "investigation skill", having an "open mind" and an "intention to learn from experience" seem to relate to the *Openness to Experience(O)* trait in the model. We explicitly sought opinions on whether the Big Five traits were common in good software testers (Section III-F), but it was not possible to mimic the elaborate assessments used by personality psychologists in a brief survey question, and the responses to this question were therefore inconclusive. da Cuhna and Geathead [8] also found a connection between personality traits and debugging ability. As they used a different assessment method with a different underlying description of personality traits, it is difficult to directly compare our observations to their results. However, we believe our results provide sufficient justification for further investigation of the connection between personality traits and the effectiveness of a software tester.

Implications of these findings for recruitment, training and management of software testers include a need to identify and support characteristics often mentioned, such as curiosity, tenacity, thinking outside the box, and ability to creatively and deliberately go about trying to "break" a system. While some of these characteristics are generally recognised to be useful for software developers in general, some are quite counter-intuitive. Testing is the only software development activity that actively and explicitly tries to "break" or crash / destroy a system - it is fundsmentally a "destructive" set of tasks. In contrast, all other software development tasks are "constructive", or at least support constructive activities. Personality or other personal characteristics that enable testers to act in a destructive mind-set may impact their effectiveness for these activities. Creative ways of testing systems not envisaged by the developers may identify weaknesses that a "constructive" mindset may not.

Many respondents mentioned communication and interpersonal skills (ability to accurately describe faults, persuade developers as to their existence and importance, and to teach them how to avoid such faults in future) as important for software testers. This was done in the context of qualities that influence performance, and, interestingly, as a way to assess their performance. Again, such skills are at least partly innate, and are generally associated with Extroversion in the Big Five model. Furthermore, if they are systematically taught at all, they are not taught in the context of specialized training in software testing tools and techniques.

This implies that both education and training of software testers may benefit from more explicit attention to these areas. Similar capabilities and characteristics are of course important for software developers in general. However, testers again have a special role in that they need to deeply understand domain and technical areas in order to be able to effectively test and understand test results. They must also effectively convey to developers and others these results and their implications. However, as we have noted, the qualities identified above are, to our knowledge, not generally the focus of specialized testing training. Indeed, at first glance, they appear to have much in common with the "generic skills" required of software developers and many other IT professionals. It is therefore an open question whether there are characteristics which are related with being an effective software tester, that are *not* as significant for other IT occupations. One quality mentioned above that may be more significant to testing than other software development activities is "investigation skill". Determining whether investigation skill, or some other quality, is *uniquely* significant for effective software testing is worthy of further examination.

As shown in Section 3.4, the majority of survey respondents did use some kind of test automation tool. They sped up testing, improved accuracy, and freed software testers to devise new tests rather than conducting repetitive and prosaic tasks. As Kaner put it, the tools extended human reach [3]. 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 software 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.* [19], 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 later that 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?

In general, our respondents tend to believe that education 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 IV-A. 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 [20], [21], 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. 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, was frequently mentioned.

Our results are consistent with our original conjecture that human factors are crucial in software testing. This supports the idea that identifying good software testers, and identifying individuals with the potential to be good software testers, is potentially valuable. For the measurement of existing software 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. Instead, producing high-quality bug reports, and the ability to communicate them effectively to developers, were considered better measures. While it was mentioned by only one respondent, the ultimate measure of a testing process is the reliability of the delivered system. It is an appealing measure, but teasing out the influence of an individual software tester on this would be challenging in a research context, let alone an industrial one. In any case, it is difficult to conclude much from our survey in this area. It is even more the case in terms of the question on prediction, where responses were insufficiently strong to conclude much at all. One interesting result was the response indicating that good programming skills do not help to improve software testing effectiveness. It is difficult to know what respondents meant by this. Many respondents mentioned that software testers should have good scripting skill and expertise in automation, which in our view is a form of programming. Only half of respondents at least somewhat agreed that a person's academic record was a good predictor of their software testing ability. This is a potentially fruitful area for further investigation; some of the qualities identified earlier in the discussion, such as intelligence and personality characteristics, may be good places to begin such an investigation.

VI. Conclusions and Future Work

Our survey results indicate that software testers do indeed think that testing-specific tools and techniques are an important part of being a good software tester. But they are only a part. Factors like intelligence, dedication, interpersonal skills, and motivation were viewed as crucial in being an effective software tester. As such, according to our respondents, the answer to the question - does the individual matter in software testing - is very clearly yes.

Therefore, it is important to identify the origins of these qualities, and how they can be measured and predicted. To what extent are they innate? What is the contribution from the general work and life experience of the software tester? Can they be taught to some extent? And, importantly, which ones are unique to software testing, and which are common to software engineers, or even IT professionals in general? Would someone with stronger traits in one or more areas likely make a better software tester, or not? Do different areas of software testing emphasise or drawn upon different personal capabilities and characteristics?

Many of these questions are the domain of applied psychology. Others, specific to software testing, can best be studied by further, more detailed investigations, rather than broad-ranging surveys. Our study raises many important questions, opening up an entirely different perspective on improving the industrial practice of software testing. As such we believe further investigation in this area will be very valuable.

Key future investigations we plan to carry out include detailed personality profiling of software professionals using the Five Factor Model to see if there appears any specific traits different between testers and other IT professionals. We want to determine better ways of assessing software tester performance, in order to be able to correlate personality and other characteristics with performance in software testing. A better characterisation of software testing tasks is necessary as it appears different tasks require quite diverse skill and expertise and potentially are impacted by different personal characteristics. Finally, investigating in more detail training and certification needs of software testing professionals and organisations would provide guidance on development of these areas.

References

[1] Beizer, Boris, *Software testing techniques (2nd ed.)*. New York, NY, USA: Van Nostrand Reinhold Co., 1990.

[2] Bertolino, Antonia, "Software testing research: Achievements, challenges, dreams," in *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 85–103.

[3] B. J. Kaner C. and P. B., *Lessons Learned in Software Testing : A Context-Driven Approach*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[4] P. G. Armour, "The unconscious art of software testing," *Communications of the ACM*, vol. 48, no. 1, pp. 15–18, 2005.

[5] J. Bach, *Exploratory testing Explained*, E. v. Veenendaal, Ed. The Testing Practitioner. UTN Publishers, 2002, www.satisfie.com.

[6] Beer, Armin and Ramler, Rudolf, "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.

[7] M. A. McDaniel, F. L. Schmidt, and J. E. Hunter, "Experience correlates of job performance," *Journal of Applied Psychology*, vol. 73, no. 2, pp. 327–330, 1988.

[8] A. D. D. 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.

[9] M. Denscombe, *The good research guide for small-scale social research projects*. Milton Keynes, UK: Open University Press, 2003.

[10] S. Boslaugh and P. A. Watters, *Statistics in a Nutshell*. Sebastopol, CA, USA: O'Reilly, 2008.

[11] N. Haslam, *Introduction to Personality and Intelligence*. London, UK: SAGE Publications Ltd, 2007.

[12] T. Kanij, R. Merkel, and J. Grundy, "A preliminary study on factors affecting software testing team performance," in *ESEM*, 2011, pp. 359–362.

[13] "A survey of key factors affecting effectiveness of software testing professionals," http://www.benambra.org/survey/.

[14] "ISTQB homepage," http://www.istqb.org/.

[15] M. J. Ree and J. A. Earles, "Intelligence is the Best Predictor of Job Performance," *Current Directions in Psychological Science*, vol. 1, no. 3, pp. 86–89, June 1992.

[16] D. C. McClelland, "Intelligence is not the Best Predictor of Job Performance," *Current Directions in Psychological Science*, vol. 2, no. 1, pp. 5–6, February 1993.

[17] A. R. Jensen, *The g Factor*. West Port, CT, USA: Praeger Publishers, 1998.

[18] D. P. McAdams, *The Person: An Integrated Introduction to Personality Psychology*, 3rd ed. Orlando, FL, USA: Harcourt, Inc, 2001.

[19] 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.

[20] Leonard L. Tripp, "Software certification debate: Benefits of certification," *Computer*, vol. 35, pp. 31–33, 2002.

[21] Adam Kolawa, "Software certification debate: Certification will do more harm than good," *Computer*, vol. 35, pp. 34–35, 2002.