The Journal of Systems and Software 86 (2013) 1354–1376

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A survey of software testing practices in Canada

Vahid Garousi [a,b,*], Junji Zhi [c]

[a] Software Quality Engineering Research Group (SoftQual), Department of Electrical and Computer Engineering, University of Calgary, Calgary, Alberta, Canada[1]
[b] Graduate School of Informatics, Middle EastTechnical University (METU), Ankara, Turkey
[c] Software Quality Engineering Research Group (SoftQual), Department of Computer Science, University of Calgary, Calgary, Alberta, Canada[1]

ARTICLE INFO

Article history:
Received 1 May 2012
Received in revised form
18 November 2012
Accepted 23 December 2012
Available online 21 January 2013

Keywords:
Survey
Software testing
Industry practices
Canada

ABSTRACT

Software testing is an important activity in the software development life-cycle. In an earlier study in 2009, we reported the results of a regional survey of software testing practices among practitioners in the Canadian province of Alberta. To get a larger nationwide view on this topic (across Canada), we conducted a newer survey with a revised list of questions in 2010. Compared to our previous Alberta-wide survey (53 software practitioners), the nation-wide survey had larger number of participants (246 practitioners). We report the survey design, execution and results in this article. The survey results reveal important and interesting findings about software testing practices in Canada. Whenever possible, we also compare the results of this survey to other similar studies, such as the ones conducted in the US, Sweden and Australia, and also two previous Alberta-wide surveys, including our 2009 survey. The results of our survey will be of interest to testing professionals both in Canada and world-wide. It will also benefit researchers in observing the latest trends in software testing industry identifying the areas of strength and weakness, which would then hopefully encourage further industry-academia collaborations in this area. Among the findings are the followings: (1) the importance of testing-related training is increasing, (2) functional and unit testing are two common test types that receive the most attention and efforts spent on them, (3) usage of the mutation testing approach is getting attention among Canadian firms, (4) traditional Test-last Development (TLD) style is still dominating and a few companies are attempting the new development approaches such as Test-Driven Development (TDD), and Behavior-Driven Development (BDD), (5) in terms of the most popular test tools, NUnit and Web application testing tools overtook JUnit and IBM Rational tools, (6) most Canadian companies use a combination of two coverage metrics: decision (branch) and condition coverage, (7) number of passing user acceptance tests and number of defects found per day (week or month) are regarded as the most important quality assurance metrics and decision factors to release, (8) in most Canadian companies, testers are out-numbered by developers, with ratios ranging from 1:2 to 1:5, (9) the majority of Canadian firms spent less than 40% of their efforts (budget and time) on testing during development, and (10) more than 70% of respondents participated in online discussion forums related to testing on a regular basis.

1. Introduction

Software testing is an important and costly activity in the software development life cycle. Furthermore, inadequate software testing usually leads to major risks and consequences. For example, a 2002 report (Tassey, 2002) by the American National Institute of Standards and Technology (NIST) reported that the negative economic impacts of lack of software testing infrastructure in the United States alone amounts to $62 billion USD per year. Needless to say, there are similar challenges in other countries.

An international survey of software engineering practices sponsored by Industry Canada in 2002 (Rubin et al., 1995) found that "Canada's software, as measured by defects per KLOC (thousand lines of code), has the highest defect rate observed". The study also noted that "Canada's work profile shows the highest share of resources being focused on defect correction". According to that survey (Rubin et al., 1995), the software produced by the Canadian software industry is rated at the low end of the software quality spectrum among the western nations, and that it requires considerable effort in defect fixing (Geras et al., 2004).

To achieve global success, Canadian firms will have to meet or exceed quality standards of leading software publishers, especially in the e-commerce era. Typical web-enabled e-commerce application teams have to deal with rapidly changing Internet

* Corresponding author at: Software Quality Engineering Research Group (SoftQual), Department of Electrical and Computer Engineering, University of Calgary, Calgary, Alberta, Canada.
E-mail addresses: vgarousi@ucalgary.ca (V. Garousi), zhij@ucalgary.ca (J. Zhi).
[1] http://www.softqual.ucalgary.ca.

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.12.051

infrastructure as well as fast turnaround times and short release intervals. In particular, the rising expectations of software consumers will require higher quality standards (Geras et al., 2004). The gap between where the Canadian software industry is now and where it needs to be still seems to be quite large. Although the above scenario describes the Canadian perspective, most leading nations face a similar dilemma. To succeed internationally, they need a strong Information Technology (IT) sector.

The IT sector worldwide faces enormous challenges in delivering quality products on time and within budget. The Standish Group is a well-known market research firm in Information Technology project and value performance analysis, which regularly publishes a report series called CHAOS on the topic. According to the 2001 CHAOS report (The Standish Group, 2001) which surveyed a very large pool of IT projects, only 28% of those projects were successful (delivered on time, on budget, with required features and functions). 23% of them failed (i.e., canceled prior to completion or delivered and never used), and 49% were "challenged" (i.e., late, over budget, and/or with less than the required features and functions). The situation in 2009 has only gotten worse, according to the latest 2009 CHAOS report (The Standish Group, 2009): only 32% of the IT projects being successful, 44% failed, and 24% challenged. Indeed, software testing and quality assurance is the major keystone in deciding the success or failure of the software projects.

An April 2002 survey by InformationWeek reported that 97% of the 800 IT staff had problems with software defects, and nine out of 10 had experienced higher costs, lost revenue, or both as a result of those problems (Hayes, 2002). On top of this, surprisingly, software testing currently consumes, on average, about 80% of the total cost of software development (Tassey, 2002). This relatively grim view of the software industry leads us to question our methods and practices. This study focuses on software testing methods and practices in particular, given the increasingly critical role of testing in newer software development methods.

In an earlier study in 2004, our colleagues Geras et al. (2004) described the results of a regional survey of software testing and software quality assurance techniques in practice, in the Canadian province of Alberta. Five years after that first study, we replicated their survey in 2009 and analyzed the change in trends from 2004 to 2009 and published the results in an article in the Journal of Systems and Software (JSS) (Garousi and Varma, 2010). To get a larger nationwide view on this topic (Canada), we conducted a newer survey with an updated list of questions in 2010. We report in this article the design, execution and results of our latest 2010 Canada-wide survey.

The remainder of this article is structured as follows. A survey of the related work is presented in Section 2. We describe the design of the survey and its execution in Section 3. In Section 4, we present and analyze the survey's results. Section 5 summarizes the findings and discusses the lessons learned. Finally, in Section 6, we draw conclusions, and suggest areas for further research.

2. Related work

A large number of surveys have been conducted on the subject of software testing practices in different countries and scales. Our literature search has identified a number of studies (Geras et al., 2004; Garousi and Varma, 2010; USGAO, 1983; Gelperin and Hetzel, 1988; QAI, 2002; Andersson and Runeson, 2002; Runeson et al., 2003; Torkar and Mankefors, 2003; Ng et al., 2004; Taipale et al., 2005; Wojcicki and Strooper, 2006; Runeson, 2006; Grindal et al., 2006; Martin et al., 2007; Ambler, 2009; Engström and Runeson, 2010; Causevic et al., 2010), which have been summarized in Table 1 (sorted by their dates) and discussed briefly next.

It is interesting to see that most of these surveys have been conducted in the last decade (since 2002), denoting the emergence of the need for surveys of these types as the software testing industry is becoming more mature. From the 16 surveys in this list, the Swedish software community, with five articles, is the most active in conducting and reporting surveys on software testing practices. The American researchers were the earliest to report the first two survey results in 1983 and 1988.

Table 1
A summary of surveys on the subject of software testing practices.

Paper reference	Scale/region	Year	Number of respondents	Goal/focus area
USGAO (1983)	USA	1983	207	Test procedures
Gelperin and Hetzel (1988)	Washington, USA	1988	Not reported	Test methodologies, test process models, growth of professionalism
QAI (2002)	India	2002	Not reported	Test standards/procedures, hardware platform, test practice, training, metrics, test certifications
Andersson and Runeson (2002) and Runeson et al. (2003)	Sweden	2003	11 Swedish companies	Test processes
Torkar and Mankefors (2003)	Sweden	2003	91	General software testing (and reuse)
Geras et al. (2004)	Province of Alberta, Canada	2004	60	Test practices, test types/levels, techniques, automation, metrics, management, training
Ng et al. (2004)	Australia	2004	65	Testing techniques, tools, metrics, standards
Taipale et al. (2005)	Finland	2005	18 experts	Ranking research directions in testing
Wojcicki and Strooper (2006)	Australia	2006	35	Context of verification and validation technology, decision process for V&V use, case study information and cost-effectiveness
Runeson (2006)	Sweden	2006	17	Unit testing
Grindal et al. (2006)	Sweden	2006	12 software organizations	Testing maturity
Martin et al. (2007)	UK	2007	A software firm in the UK	Integration testing, requirements testing, test coverage, automating testing, test scenario design,
Garousi and Varma (2010)	Province of Alberta, Canada	2009	53	Test practices, test types/levels, techniques, automation, tools, metrics, management, training
Ambler (2009)	Online testers community	2009	121	The adoption rate of Test-Driven Development (TDD), TDD related techniques
Engström and Runeson (2010)	Sweden	2010	32	Regression testing
Causevic et al. (2010)	Europe	2010	83	Contemporary aspects of software testing
The current survey	Canada	2010	246	Test practices, test types/levels, techniques, automation, tools, metrics, management, training, research and interaction with academia

The works in USGAO (1983) and Gelperin and Hetzel (1988) are considered two of the early works on the topic. The work in USGAO (1983) is a public report by U.S. general accounting office to the US government's administrator of general services and is titled: "Greater Emphasis on Testing needed to Make Computer Software more Reliable and Less Costly". In this report, 207 staff members of the US government were surveyed and a comprehensive report was produced. Among the findings were that the agencies do not always enforce testing policies and requirements. For example, the required unit and system testing for a payroll modification was omitted because the programmer considered the change minor.

Gelperin and Hetzel (1988) reported the growth and industry penetration of software testing. Their survey was filled out by the industrial participants of the 4th International Conference on Software Testing held in Washington, DC on June 1987. One of the study's interesting observations is that: "*Testing, like everything else, can be either underdone or overdone*". The most commonly reported practice was recording of defects found during testing (73% of the respondents). The least common practice was measuring code coverage (e.g., number of executable source lines executed) achieved during testing (only 5% viewed this as a normal practice). A high percentage of the respondents felt that the testing in their organization was a systematic and organized activity (91% answered yes). It should be understood that the results of that survey (Gelperin and Hetzel, 1988) are strongly biased. People who attend technical conferences represent the leading edge of test awareness and sophistication. Therefore, the survey does not measure general industry practice but the practices of that small part of the software development community that is aware of the issues and wants to do improve it. Some observers believe that general industry practice is much worse than the survey profile. This view is supported by comparing the statistics to an earlier 1983 study by the US Government's accounting office on software testing practices in federal agencies (USGAO, 1983).

Another survey report (QAI, 2002) is compiled by the Quality Assurance Institute based on a survey completed at its international conference on software testing in India in 2002. Similar to the above-mentioned survey (Gelperin and Hetzel, 1988), the results of this survey are biased, since conference attendees are presumably software testing professionals who recognize the value of testing.

A qualitative survey was conducted in 2003 to investigate the test process practices in software industry (Andersson and Runeson, 2002; Runeson et al., 2003). After analyzing the responses from eleven Swedish companies, the authors found that companies had various attitudes toward the value of test process. Larger organizations emphasized documented development process as a key asset while smaller organizations relied more on senior practitioners. In terms of evolution approach, incremental and daily build are reported as two most popular approaches. Finally, the authors identified two types of test automation: recorded data automation and scripting automation. According to the conclusion, products which focus on non-functional properties are primarily tested using recorded data. In contrast, scripting automation technique is mainly used to test products with a focus on functionality.

Torkar and Mankefors (2003) studied the testing activities in the context of code reuse. 91 Sweden-based participants from open-source projects and three companies contributed to the survey. The survey results showed that that a majority of the developers participating in the survey did not test reused code and other testing methodologies were not used to the extent that the scientific community takes for granted. A more automated approach to testing in combination with code coverage analysis and statistical analysis was found to be needed.

Ng et al. (2004) studied the software testing practices in Australia during years 2003 and 2004. 65 industrial respondents completed the survey. The study finds the most evident barriers to using software testing methodologies and techniques to be a lack of expertise among the practitioners. This finding suggests that there could be a vast number of software testing staff who are not appropriately trained in the use of formal testing methodologies or techniques. This may indicate a deficiency in the training of software testing professionals to meet the actual demand of the industry, or deficiencies in the techniques themselves. Cost was ranked first in the list of barriers to the use of automated testing tools.

The initial prediction of Ng et al. (2004) is that government and public non-commercial organizations, being publicly funded, are very likely to adopt structured testing methodologies. However, to their surprise, they found that while there are about 70% of private organizations adopting some form of structured testing methodology, government and public organizations reported a significantly lower percentage. About 75% of organizations allocated less than 40% of their development budget to software testing activities and there was a substantial level of satisfaction among organizations that hired external testers to assist them in software testing activities. The most popular type of tools used is to support test execution (35 out of 44 organizations), followed by regression testing (33 organizations), with result analysis and reporting tools (27 organizations) being the third.

Taipale et al. (2005) invited 18 experts from industry and academia to evaluate the issues in software testing research. The top three research issues ranked in this survey are: (1) test process improvement, (2) testing automation and testing tools, and (3) standardization. The authors took a further step to propose a hypothesis for future research. According to the authors, problems in the information flow between software development and testing processes may increase both development work and testing. However, the author admitted that this survey results had a low applicability as they can only be applied to similar environments.

Ratios from another survey (Wojcicki and Strooper, 2006) on verification and validation for concurrent programs, conducted by two Australian researchers, are worthy of notes. There were 35 worldwide respondents in this study (Wojcicki and Strooper, 2006) who were selected from the IBM developer works multithreaded Java programming forum. It is interesting to observe that most respondents in the above-mentioned survey (QAI, 2002) conduct system testing, while most of the respondents in this survey (Wojcicki and Strooper, 2006) reported conducting unit testing.

Runeson (2006) surveyed unit testing practices on the basis of focus group discussions in a software process improvement network (SPIN). Their goal is to investigate the meaning of unit testing for software practitioners. As their survey results showed, the participants agreed on the scope of unit testing but disagree on whether test environment should be a separate software system. Moreover, managers or Q/A teams left unit testing as solely developer issues and place little impact on test strategies or practices. In terms of test coverage, it is discouraging to see that developers rarely measured structural coverage.

Grindal et al. (2006) reported an interview study on the test maturity of software organizations. Test managers from twelve European organizations were recruited. The data indicated that, although the test maturity is low due to lack of proper structured test strategies, these organizations are commercially successful. Many interviewed managers expressed concerns over low test maturity which indicated their awareness of the problem. The author finally proposed a suggestion to the managers that they concentrate on a metrics program so as to allow the economic-based management of product quality.

Martin et al. (2007) reported a UK-based case study from a small software house. To illustrate their findings, the authors described in detail an example of integration testing in this firm. Through the example, they identified some organizational realities and

constraints that shaped testing-related aspects such as test coverage, requirement testing, test scenario design, etc. Finally, the authors concluded that there was a disconnect between testing research and practices and pointed out the need to address the relationship between the two.

A recent survey about Test-Driven Development (TDD) was conducted in 2008 and its results are published in Ambler (2009). The survey was announced on the Extreme Programming (XP) and Test-Driven Development (TDD) mailing lists and there were 121 respondents. The goal was to compare how the actual practices used by agile developers compared with the process model advocated in the agile literature. Some of the main findings of the survey are the followings. Within the TDD/XP community, there is a significant focus on other testing and validation techniques, such as inspections, regression testing, and end-lifecycle testing. Even though there is a clear bias toward TDD on these communities, TDD is clearly not the only validation technique that has been commonly adopted by agile developers. Developer TDD is more common than acceptance TDD (Koskela, 2007), perhaps because of the more technical focus within this community. For both types of TDD (developer and acceptance), pairing with an experienced person and mentoring were rated as the most effective means of learning the technique. Training came in a distant third followed by other techniques such as pairing with another learner, reading, and online forum discussions. The survey data (Ambler, 2009) provides evidence to support the claim that although many groups claim to be using acceptance TDD (Koskela, 2007), it is still more common for them to capture requirements specifications in the form of documents or models.

Regression testing has gain substantial attention both in academia and industry. A more recent survey was reported on this topic in 2009 (Engström and Runeson, 2010). What is unique in this qualitative survey is that, the authors first ran a focus group meeting with 15 industry participants and then validated the outcome in an online questionnaire from 32 respondents. In term of the definition of regression testing, most respondents reached a consensus on a formal meeting. However, regression test practice differed in different contexts, varying in scope and preconditions. On the other hand, most challenges highlighted in the study were not specific to regression testing but are general to other testing types. Among these challenges, the authors pointed out that test automation and friendly test environment support have significant impact on the effectiveness of regression testing practices.

Causevic et al. (2010) reported an industrial survey which focused on perceptions of the software testing process. Before analysis the authors divided the respondents into four different categories: safety-criticality, agility, distribution of development, and application domain. Their findings reveal notable discrepancies between preferred and actual testing practices. One of the noteworthy results from the quantitative analysis on satisfaction level of practitioners is related to TDD development. Out of five problem statements, TDD was noted where the difference between the preferred practice and the current practice is most significant.

We should mention that the existing surveys in this area contained a mix of quantitative and qualitative questions. In quantitative questions, exact quantitative answers (e.g., numbers or percentage values) are reported by participants. On the other hand, in qualitative survey questions, questions of qualitative nature are asked from participants, e.g., "what are your challenges in testing your software?" Usually, analysis, synthesis and reporting of qualitative survey results are more challenging than quantitative surveys. In the pool of surveys discussed above, three of the surveys had a strong emphasis on qualitative surveying techniques (Andersson and Runeson, 2002; Runeson et al., 2003; Engström and Runeson, 2010), which were discussed above.

The two studies (Geras et al., 2004; Garousi and Varma, 2010) were conducted by us and our colleagues to investigate regional Alberta-wide survey of software testing and quality assurance techniques in practice in 2004 and in 2009. More details about the 2004 and 2009 survey are discussion in Section 3.1.

3. Survey goal, design and execution

The survey goal and research questions are presented in Section 3.1. We then present the design of the survey in Section 3.1. Section 3.2 discusses the survey execution and the statistics on the respondents' population. For conducting this survey and reporting its results, the authors have benefited from the case-study research guidelines presented by Wohlin et al. (Runeson and Höst, 2009; Wohlin et al., 2000).

3.1. Survey goal and research questions

The approach we used in our survey study is the Goal, Question, Metric (GQM) methodology (Basli, 1992). Using the GQM's goal template (Basli, 1992), the goal of our survey is to characterize the software testing practices in Canada for the purpose of identifying the trends, and also to provide a view on the latest testing techniques, tools and metrics used by professional testers and challenges faced, to be able to benefit both testing professionals and also researchers both in Canada and world-wide, for observing the latest trends in software testing industry and identifying the areas of strength and weakness and encouraging more academia–industry collaborations. Based on the above goal, we raise the following research questions (RQ).

RQ1. How much effort is spent by the industry on each type/level of testing (e.g., unit, integration, and system testing)?
RQ2. What are the most- and least-used test techniques?
RQ3. How much test automation is practiced and what specific test tools are used most often?
RQ4. What types of test metrics are used?
RQ5. How are the test teams and processes managed?
RQ6. How much internal and external training on testing is provided in companies?
RQ7. How much in-house research and also interaction with academia is conducted in firms?

Each of the above RQs is used to derive several "survey questions" and metrics in the following section.

3.2. Survey design and questions

In our 2009 survey (Garousi and Varma, 2010), since our goal was to compare the Alberta-wide testing practices in year 2009 to the data from an earlier 2004 survey (Geras et al., 2004), we had mostly reused the questions from the 2004 survey, and had added four additional questions. However, since both the software testing research literature and practice has continuously evolved since 2004, the authors decided to come up with a "fresh" list of questions for the Canada-wide survey in 2010.

In order to develop a survey that would adequately cover the latest topics in software testing while at the same time permitting us to economize on the number of questions, we reviewed the similar past surveys (Geras et al., 2004; Garousi and Varma, 2010; USGAO, 1983; Gelperin and Hetzel, 1988; QAI, 2002; Andersson and Runeson, 2002; Runeson et al., 2003; Torkar and Mankefors, 2003; Ng et al., 2004; Taipale et al., 2005; Wojcicki and Strooper, 2006; Runeson, 2006; Grindal et al., 2006; Martin et al., 2007; Ambler, 2009; Engström and Runeson, 2010; Causevic et al., 2010) and

designed a draft set of questions. The authors then consulted with several industrial practitioners to do a careful peer review on the draft set of questions. Getting feedback from industrial partners in design of surveys is an approach followed in previous surveys as well (e.g., Torkar and Mankefors, 2003). The goal behind this phase in our survey design was to ensure that the terminology used in our survey was familiar to a reasonable ratio of the audience. This is since, unfortunately, the software testing terminology used in academia versus industry can sometimes be slightly different or even confusing, e.g., system testing and functional testing. The useful feedbacks from the industrial practitioners were used to finalize the set of survey questions.

Similar to the earlier 2004 and 2009 studies (Geras et al., 2004; Garousi and Varma, 2010), we benefited from the Software Engineering Body of Knowledge (SWEBOK) (Abran et al., 2001) in categorizing our questions. SWEBOK divides the software testing knowledge area into four categories: (1) test levels, (2) test techniques, (3) test-related measures, and (4) test process management. Based on our original survey motivation and also in consultation with our industrial contacts, we also added three additional categories: test automation and test tools, test training, and research/interaction with academia.

The criteria we used in designing our survey's question set were to make sure that the questions are relevant to the industry and also to capture the most useful information. The complete list of the 34 questions used in the 2010 survey is shown in Table 2. Note that, for each of the seven RQs discussed in Section 3.1, several "survey questions" and their corresponding metrics have been derived. To enable traceability among the RQs and "survey questions", the RQ numbers have been labeled under the first column in Table 2.

Most of our questions had quantitative pre-designed multiple-choice answers (e.g., the 1st question regarding the participant's current position), while a few had qualitative (free text) answers, e.g., "What challenges would you like to raise for the research community?" For brevity, we are not showing the pre-designed multiple-choice answers of the survey's questions in Table 2, but they can be found in the authors research group website (Garousi and Zhi, 2012). Since we will be comparing the 2010 survey results with the similar questions of the 2004 and 2009 Alberta-wide surveys, the questions used in the 2004 and 2009 Alberta-wide surveys are also included in our website (Garousi and Zhi, 2012), for referencing and comparison purposes.

The first category of questions (#1–11 in Table 2) collected respondents' profiles. The second category of questions (#12–#14) collected test type/level-related data. Test types/levels refer to the stages of testing (unit, integration, and system testing) as well as the ability of the test to assess certain properties of the system under test. Our survey aimed to determine the extent to which companies are using the test levels, and to identify the product characteristics that they are currently testing for.

The 3rd category of questions (#15 and #16) collected the information about test techniques. Test techniques are mechanisms for identifying test cases and include a wide variety of methods, from ad hoc to formalized call graphs. The survey question #15 sought to identify the test techniques that software organizations are using (and the ones they are not using). We also had a question (#16) about mutation testing (fault injection) in specific since, according to our experience, few testing practitioners know about it or use it (even if they are familiar with it).

The two questions in the test automation and test tools category aim at collecting the level of test automation (#17) and also specific test tools used in companies (#18). Test metrics being used by organizations were also investigated (questions #19 and #20). Our goal was to collect test coverage metrics (e.g., line and decision coverage) and also project- or company-wide software quality metrics (e.g., defect per line of code, defects detected per day). The extent to which these metrics were used is important given that Agile methods such as Scrum and Extreme Programming (XP) rely on measurements to inform the approach to subsequent iterations. Scrum, for example, is positioned as an "empirical" method for "controlling chaos" (Schwaber et al., 2001). Extreme programming uses "yesterday's weather" in order to assist the team in detailed iteration planning.

We also sought to understand the ways in which organizations manage their testing processes (questions #21–#24). Recent shifts in industrial best practices make testing even more pervasive in the development process. It makes sense that managing the test effort is also increasingly important. The survey assessed this aspect with a number of questions related to managing the test process.

The next category of questions was test training (questions #25–#27). The rationale of these three questions were: (1) to understand the level of formal training testing practitioners receive on the subject of testing, (2) if organizations have a specific training program for testing, and (3) barriers preventing companies from offering software testing training.

We also wanted to measure the extent of research on software testing inside companies (e.g., by having corporate R&D centers), their interaction with the university researchers (academia) and reading technical articles on testing and attendance in software testing conferences. Questions #28–#34 served for this purpose.

3.3. Survey execution

They survey was designed and hosted on an online survey hosting service called *SurveyMonkey*[2] and was available to the invitees for three months during the summer of 2010. Research ethics approval for the survey was obtained from the University of Calgary's Conjoint Faculties Research Ethics Board (CFREB) in April 2010 and then the survey was executed. Participants were asked to complete the survey online. Respondents could withdraw their results from the survey at any time and, as per the ethics guidelines, researchers agreed to publish only summary and aggregate information from the survey.

To ensure that we would receive as many as responses as possible, we followed the following survey invitation strategy. In addition to the email list of companies used by the two earlier studies (Geras et al., 2004; Garousi and Varma, 2010), we sent email invitations to our network of partners/contracts in Canadian software companies. We also sent email to general email addresses of major Canadian software companies that we knew about, but did not have any contacts in, e.g., Research In Motion (RIM) Limited.

The sample included companies with development offices in Canada, whether those offices were the company's head office or not. Since development offices (conducting software development and testing tasks) were of interest to us in studying their testing practices, whether or not the company had a head office in Canada was not an important factor in our study. The same criterion was also applied in the two earlier surveys (Geras et al., 2004; Garousi and Varma, 2010) as well.

3.4. Respondents population and sample size

In total, we received close to 300 responses. As another step to ensure the quality of our data, we analyzed the IP addresses of the respondents and filtered out the replies from non-Canadian IP addresses. They were a few records which were filled out from India and a few other countries. At the end, our survey data included responses from 246 software practitioners inside Canada.

[2] www.surveymonkey.com.

Table 2
List of the questions used in the 2010 Canada-wide survey.

Aspect (RQ)	Questions (and metrics)
Respondents profiles and demographics	1. What is (are) your current position(s)? 2. How many years of work experience do you have in IT and software development industries? 3. How many years of work experience do you have in software testing, in specific? 4. Which of the following categories your organization belongs to? 5. What is your highest academic degree? 6. What is (are) your university degree(s) in? 7. What software testing certification(s) do you hold? 8. Is your current development team(s) explicitly refers to itself as Agile or traditional (CMMI-focused)? 9. Please choose the province/city of your residence? 10. What is the size of your company (number of employees)? 11. Which programming languages do you use in your company?
Test types/levels RQ1	12. In your current or most recent software project, what types of test cases and test suites did you or your team develop and use? 13. If you conduct more than one type of testing, out of the total testing time and budget, how much emphasis (in percentage) you put on each test activity (roughly speaking)? (Please make sure that the summation of all the numbers adds up to 100%) 14. In terms of the type and phasing of software development life-cycle, what is the type of your test activities?
Test techniques RQ2	15. In your current or most recent software project, what test technique did the team use to generate test cases? 16. Are you familiar with Mutation Testing (fault injection to find faults)? If yes, how often do you use it in your testing projects?
Test automation and test tools RQ3	17. Overall in all of your past projects, how much automated versus manual testing have you done? (Please make sure that the summation of the two numbers should add up to 100%) 18. Which test automated tools/frameworks do you use in your company?
Test metrics RQ4	19. Which is of the following code (test) coverage metrics do you measure in your test activities? 20. Which of the following other test and quality metrics do you explicitly measure in your projects?
Test management RQ5	Please identify the ratio of testers to developers (testers/developer) in your current or most recent project. For example, if you had on average one tester for every 2 developers, the answer would be 1:2. 22. What criteria are used in your projects to decide that the testing activities are terminated/completed? 23. What barriers do you believe prevents your company from adopting systematic testing methodologies and testing tools? 24. Roughly speaking, how much budget and time is spent on testing (in percentage) across the software development process in your company? Is this just traditional testing activities or are you including unit testing here. Maybe be more specific.
Test training RQ6	25. In your entire career, have you received any formal software-testing related training (such as university or on-site training courses)? Self study based on online resources or books does not count here. 26. Does your organization have a training program that specifically targets any of the following? 27. What barriers do you believe prevents your company from providing software testing training?
Research and interaction with academia RQ7	28. In your company involved in any software testing research to develop new ways/techniques to test software systems? (Hint: conventional application of the existing test techniques is not considered research). 29. Please provide a list of top 3 testing challenges that you have been seeing in your projects and you would like the software testing research community to work on. (For example, you might say: the current record/playback GUI testing techniques/tools are not very stable and need to be improved). 30. How often do you interact (talk) to software testing university researchers about your test-related problems/challenges. 31. Which of the following software testing conferences have you attended in the past? 32. How often do you attend software testing conferences? 33. How often do you read technical papers (articles) published in software testing journals, conferences or workshops? 34. How often do you read/participate in online discussion forums related to software testing such as mailing lists, blogs or newsletters?

It is very important to calculate (or estimate) the sample size (i.e., sampling ratio) for a survey (Scheaffer et al., 2011). To estimate the sample size, we referred to a publicly available statistics of software companies and employees in Canada, which was published by Statistics Canada in 2007 (Statistics Canada, 2010) (inside Statistics Canada, 2010, refer to *Table 2: Summary statistics for the software publishers industry*). The authors acknowledge that the company size is not necessarily the best measurement in this context, and the size of the organizational unit would probably be more precise for estimating the sample size. There are often small, understaffed and under-resourced development teams even within the large corporations. But note that data for organizational unit sizes were not available in Canada (at least publically). Thus, we used company size as a rough approximation.

Further note that more recent data (after 2007) were not available from Statistics Canada. According to this report (Statistics Canada, 2010), as of 2007, there were 39.181 software engineers/developers as "paid employees" and 2156 software development establishments (firms) in Canada. Fig. 1 presents the detailed statistics of the "software publishers" industry in Canada (adapted from Statistics Canada, 2010). Data for only eight provinces (Alberta, Ontario, Quebec, Manitoba, Saskatchewan, British Columbia, Nova Scotia, and New Brunswick) have been provided in the governmental report (Statistics Canada, 2010), and no data were available for the other two provinces (Newfoundland and Labrador, Prince Edward Island) and the three Canadian territories (Northwest Territories, Yukon, and Nunavut).

Considering that 246 respondents out of the pool of 39,181 software engineers/developers in Canada contributed to this survey, a rough measure of the sample size equals 0.62% (246/39,181). In our 2009 Alberta-wide survey (Garousi and Varma, 2010), the sample size was about 1.76% (53 respondents/3000: estimated number

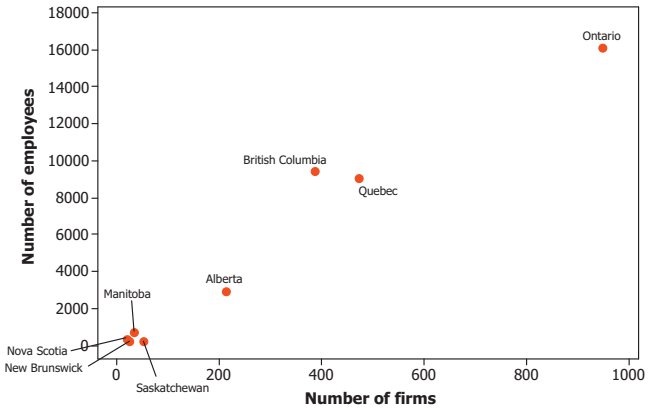

Fig. 1. Number of firms and employees in the "software publishers" industry for each Canadian province.
Data source: Statistics Canada (2010).

of software engineers in Alberta). In comparison, the number of respondents to our 2010 survey is higher than the similar studies discussed in Section 2 (Table 1). Except for those studies which did not give explicit number of respondents (refer to Table 1), most related studies had less than 200 respondents while ours had 246.

4. Survey results and findings

We report in detail the survey results and analyze its findings. The results are presented in the same order as they appeared in the survey design (Table 2).

- Profiles and demographics of the participants (Section 4.1).
- Test types and levels (Section 4.2).
- Test techniques (Section 4.3).
- Test automation and test tools (Section 4.4).
- Test metrics (Section 4.5).
- Test management (Section 4.6).
- Test training (Section 4.7).
- Research and interaction with academia (Section 4.8).

4.1. Profiles and demographics of the participants

4.1.1. Respondents positions (Q1)

The positions of respondents in this survey are shown in Fig. 2. Note that since this was a multiple-choice question, role overlap could be recorded, e.g., a person can be a developer and a tester at the same time. As the figure shows, most of the participants were testers and developers. Some participants also had higher management positions. Note that the axes showing the numbers in all

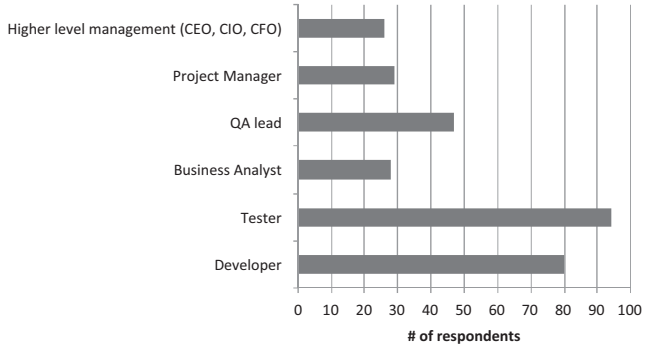

Fig. 2. Respondent positions.

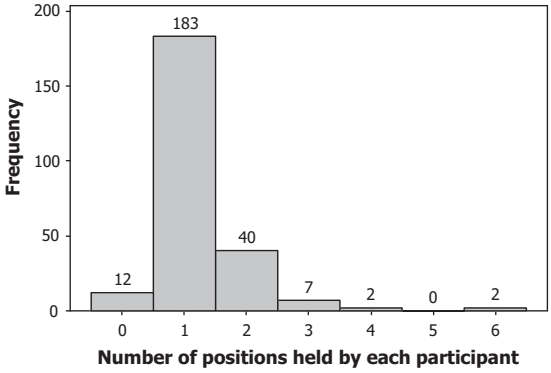

Fig. 3. Histogram of number of positions held by each participant.

figures in this section are number of participants (unless otherwise mentioned).

We should note that, as it has been established in studies on information quality (for example by Garvin, 1988), people in different positions see and rate importance of different issues differently and in general have varying viewpoints on software testing and related processes. The participants in our study were from several different roles (e.g., developers and managers).

Since this question was a multiple-choice question, we measured the frequency of the cases in which a respondent had reported more than one position. Results are shown in Fig. 3. Since all the questions were optional to be filled and not required, 12 respondents left this question unanswered ("0" positions in Fig. 3). Most of the respondents reported to be in only one position. Surprisingly, 6 respondents reported to have been assigned to all the 6 given roles concurrently.

Since this question was a multiple-choice question, we also measured the frequency of each specific joint position. Results are shown in Table 3, e.g., 6 respondents mentioned that they were developers and testers at the same time, and also developers + business analysts. The ratios of most joint positions are rather low, i.e., between 1.2% and 3.2% (3–8 respondents in each category). A relatively slightly higher number of respondents (16 of them) reported that they are both testers and QA leads, which could be expected as these two positions are inter-related.

4.1.2. Work experience in IT/software development and software testing/QA (Q2 and Q3)

Fig. 4 shows, as an individual value plot, the distribution of participants' work experience in IT and software development, and in software testing and QA. The average values are 9.5 and 4.7 years, respectively.

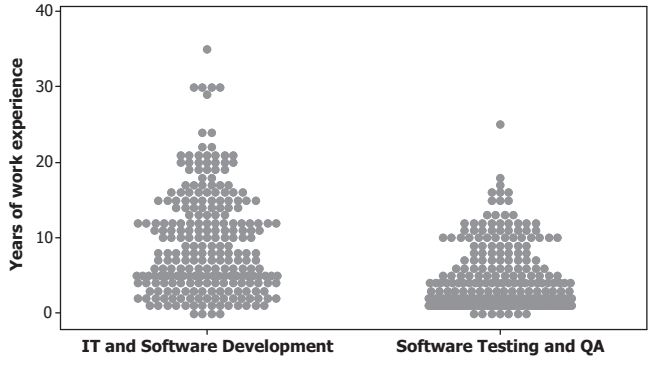

Fig. 4. Respondents' work experience in IT/Software Development and Software Testing/QA.

Table 3
Frequency of joint positions.

	Developer	Tester	Business analyst	QA lead	Project manager
Developer	–				
Tester	6	–			
Business analyst	6	5	–		
QA lead	5	16	4	–	
Project manager	6	5	8	4	–
Higher level management (e.g., CEO, CIO, CFO)	5	4	4	3	3

63.0% and 6.9% of respondents had over 5 and 20 years of working experience in IT and software development, respectively. On the other hand, only 32.1% and 0.4% of respondents had over 5 and 20 years of working experience in software testing/QA, respectively. As somewhat expected, the average duration of work experience in IT and software development is larger than that of software testing and QA, as staff members are usually start their careers in non-testing roles and then move to software testing and QA roles. Interestingly, one participant reported having more than 30 years of testing experience. We acknowledge the positive impact of diversity in participants' population as responses from participants with various lengths of work experiences can help the survey to gather valuable inputs from a wider audience base.

67.9% of respondents had less than 5 years of testing-related experience. Compared to the software developers' trend, it is interesting to observe that most testers are quite junior in their roles (that than 5 years of testing-related experience).

We also wanted to analyze whether people in higher positions (e.g., project managers and QA leads) have more years of experience compared to other roles. We clustered the data for years of work experience based on reported positions and the data are visualized as a box plot in Fig. 5. It is somewhat unexpected and surprising that no major differences among the box plots are observable. The majority of the distributions tend toward younger population (about 7–10 years of work experience). In a summary, it is not necessarily true that people in higher positions (e.g., project managers and QA leads) have more years of experience compared to other roles. This might perhaps be due to the fact that different firms have different promotion "ladders", e.g., in a typical small company, a junior software engineer with 2–3 years of experience might be promoted to the project manager position based on her/his excellent performance, which in another large-size company with more "bureaucracy", going up the ladder and reaching a project manager position might take 10–15 years. Our own personal experience also confirms this observation.

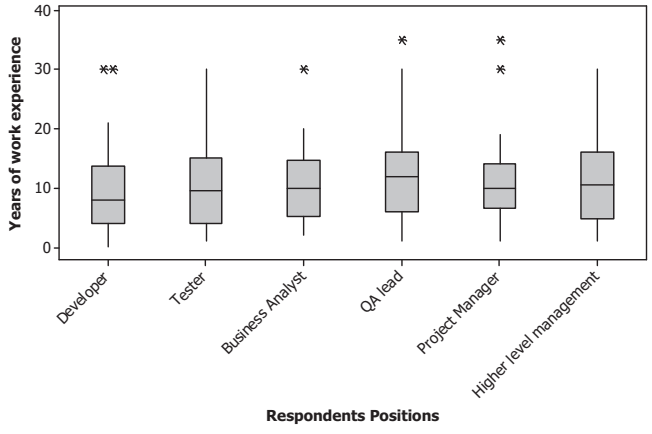

Fig. 5. Box plots of years of work experience in IT/Software Development for each position type.

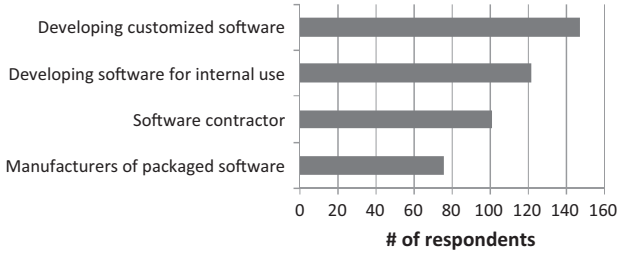

Fig. 6. Type of organization.

4.1.3. Type of organization (Q4)

The fourth question was about the type of the respondents organizations (Fig. 6). Four possible choices were pre-provided in the questionnaire, which were based on the replies we had received in our previous survey (Garousi and Varma, 2010). We can see that most respondents categorized their organization as a "customized software" development firm, followed by those who develop software for internal use. Software contractors and manufacturers of packaged software were the next ones.

Although the quality of software systems developed by each of the above organization types is important, there are often slight differences in implications of testing activities among them, e.g., the client of a customized software is usually one single firm, and thus it will have limited user base, while a packaged software is generally meant to be used by a much wider user base, thus requiring much more in-depth testing.

4.1.4. Academic background (Q5)

In order to understand the respondents' educational background, participants were asked to provide their highest academic degree. Shown in Fig. 7, the result reveals that 42.9% and 33.5% of respondents have a Bachelor's and a Master's degree respectively. 12.7% had only college degrees. The remaining 4.5% had a high school degree and lower. Surprisingly, there was no respondent with a PhD degree in our respondent pool.

Since we had the individual data records for respondents' highest academic degrees and their positions, we measured the count of pair-wise combinations, e.g., how many people have a BSc and are working as developers? The rationale was to assess whether people in higher positions (e.g., project managers and QA leads) tend to have higher degrees. The results are shown in Table 4. By comparing

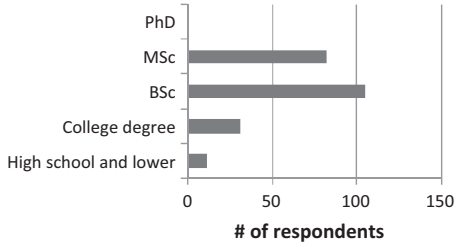

Fig. 7. Respondents highest academic degrees.

Table 4
Count of pair-wise combinations of highest academic degrees versus positions.

Highest academic degrees	Positions				
	Developer	Highest academic degrees	Business analyst	QA lead	Project manager
BSc	31	45	11	18	10
MSc	32	29	11	17	10
PhD	0	0	0	0	0
High school and lower	7	5	1	2	1
College degree	7	8	3	8	6
The degree field was not filled	3	7	2	2	2

the numbers, we cannot see any statistical significance in support of the above hypothesis. Thus, it seems to us that the conventional norm for promotion to higher positions is generally applicable in the industry, i.e., regardless of their academic degrees, staff members with proven excellence in their work would be promoted to higher positions.

4.1.5. Academic major (Q6)

As a follow-up question to the previous question, in order to understand the respondents' educational skill-set, participants were asked to provide their last academic major (e.g., software engineering, and computer engineering). The results are shown in Fig. 8.

Computer science, electrical engineering and software engineering are the top three. It is interesting that the number of participants with an electrical engineering degree is the second highest, and also the number of participants with a Business degree or an MBA is higher than those with computer engineering degrees. 28 and 11 respondents selected "other" and "none", respectively. Among the other majors were: Commerce, English, Industrial Engineering, Mathematics, and Project Management. From our personal discussion with several testers from these backgrounds, such testers are invaluable assets to their teams since they carry a wealth of knowledge in their "business domains" and could benefit the testing processes in finding business-domain-type defects (e.g., in banking applications).

4.1.6. Software testing certifications (Q7)

In this question, participants were asked what software testing certification(s) they hold. Out of all 246 respondents, only 32 respondents (about 13%) mentioned that they have at least one type of software testing certification (Fig. 9). About 87% of participants (214 out of 246) responded that they did not hold any certifications. Thus, it seems that, at least in the Canadian software testing industry, having software testing certification is not so common. The five types of certifications reported by participants were:

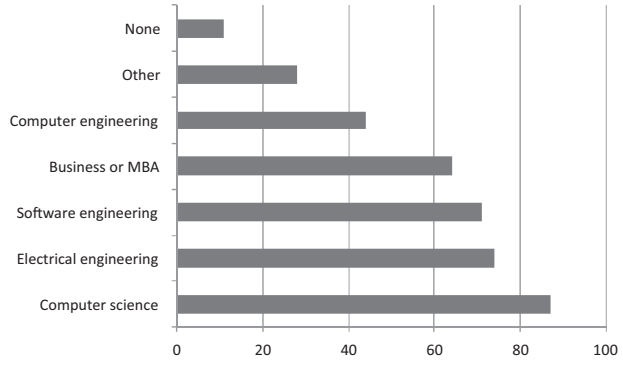

Fig. 8. Respondents academic major.

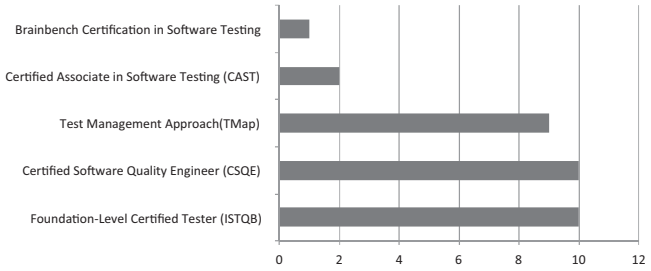

Fig. 9. Respondents testing certifications.

- Foundation-Level Certified Tester, offered by the International Software Testing Qualifications Board (ISTQB).
- Certified Software Quality Engineer (CSQE), offered by the American Society for Quality (ASQ).
- Test Management Approach (TMap), offered by an IT services company named Capgemini.
- Certified Associate in Software Testing (CAST), offered by Quality Assurance Institute (QAI).
- Brainbench Certification in Software Testing, offered by a company named PreVisor.

There are several studies on the standard for test processes and practices (ISO/IEC 29119) (e.g., Kasurinen et al., 2011a, 2011b) which have made similar observations in terms of low ratio of practicing software testers having software testing certification.

The ISTQB certification is becoming popular in the global scale, having over 240,000 certifications issued (as of June 2012). As of March 2012, ISTQB consists of 47 member boards worldwide representing more than 71 countries. The Canadian Software Testing Board (CSTB) is the Canadian national branch of the ISTQB.

4.1.7. Agile versus traditional development methodologies (Q8)

Question 8 was about the software development methodology used by participants. Among all respondents, 110 of them selected the option "Agile" whereas only 54 referred to themselves as using traditional development methodology (Fig. 10). The remaining 65 respondents did not distinguish their methodology as any explicit type. These data seem to indicate that the Agile methodologies are now surpassing the traditional approaches in terms of adoption in Canada.

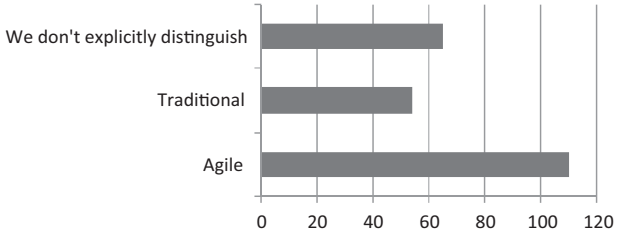

Fig. 10. Agile versus traditional development methodologies.

Table 5
Count of pair-wise combinations of company sizes versus types of development methodologies (out of 246 participants).

Company size	Types of development methodologies		
	Agile	Traditional	Not explicitly distinguished
1–5	16	9	21
11–50	47	7	28
50–500	17	18	14
500+	30	20	2
Total	110	54	65

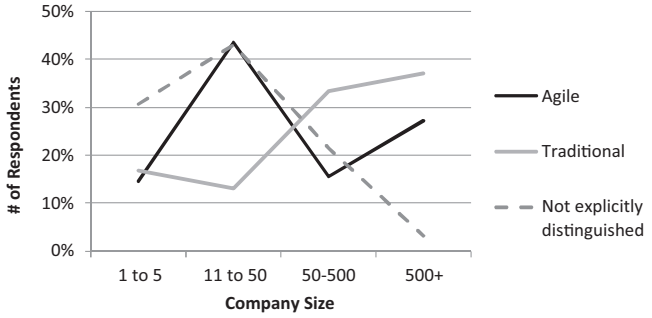

Fig. 11. Normalized data of combinations of company sizes versus types of development methodologies.

Another interesting pair-wise analysis in this context was to study the count of pair-wise combinations of company sizes versus types of development methodologies to see whether there is any relationship between the two factors, based on our survey results. The net count of each pair is shown in Table 5 (the numbers are out of 246 participants). To normalize the results for easier trend analysis, we calculated the relative percentage values (i.e., ratio of each value in Table 5 divided by the total number of responses for the corresponding type of development methodologies). For example, 14.5% and 43.6% of the participants who mentioned adapting Agile were in companies of size ranges: 1–5 and 11–50 employees, respectively. The normalized data in percentage values are visualized in Fig. 11. We can notice in this line chart a weak trend denoting that Agile is mostly adapted in small to medium-size companies (11–50 employees) compared to larger firms. On the other hand, "traditional" development methodologies seem to have been adapted more in larger firms. This interesting insight confirms the general belief regarding these two factors in the software engineering community.

4.1.8. Provincial breakdown of the respondents (Q9)

The two previous 2004 and 2009 surveys (Geras et al., 2004; Garousi and Varma, 2010) were conducted in the province of Alberta only. In the 2010 Canada-wide survey, we intended to find out the provincial breakdown of the respondents to ensure that all provinces are reasonably well represented in the data set. The results are presented in Table 6.

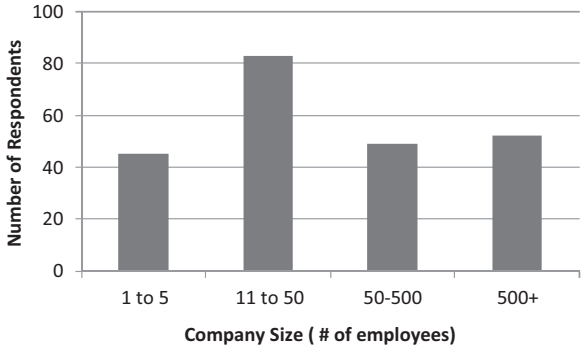

Fig. 12. Company size.

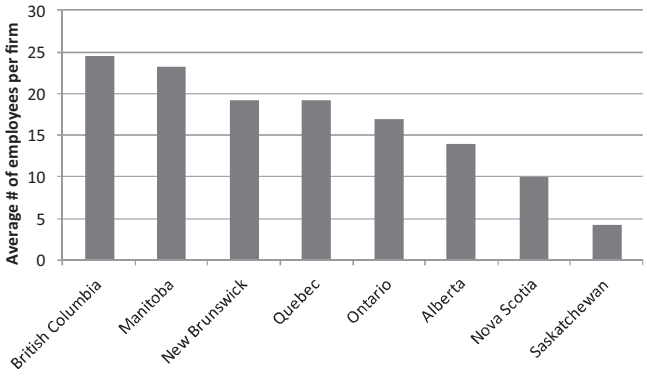

Fig. 13. Average numbers of employees per firm in the "software publishers" industry for each Canadian province.
Data source: Statistics Canada (2010).

Due to the location of the authors (the city of Calgary in the province of Alberta) and the fact that most of our contacts were local, about 44% of the respondents were from Alberta. The remaining shares of respondents were, in order, from British Columbia, Ontario, and Manitoba. Despite our efforts, there were no respondent from Prince Edward Island, New Brunswick or Newfoundland and Labrador.

4.1.9. Company size (Q10)

The histogram of company sizes in terms of number of employees is shown in Fig. 12. Most firms had between 11 and 50 employees. It is interesting that more than 52 participants from large corporations with 500+ employees also completed our survey, helping the analysis to cover a wider spectrum of inputs in terms of company sizes.

To put the above numbers in context, we calculated the average numbers of employees per firm in the software development industry for each Canadian province using the values of number of firms and employees available from Statistics Canada (2010). Results are shown in Fig. 13. This average varies from about 4 (in Saskatchewan) to about 25 in British Columbia. This seems to denote that software development firms in British Columbia are

Table 6
Provincial breakdown of the respondents and sample sizes for each province.

	Canada	Provinces							
		Ontario	British Columbia	Quebec	Alberta	Manitoba	New Brunswick	Nova Scotia	Saskatchewan
# of software firms	2156	949	385	473	212	32	18	23	52
# of software engineers	39,181	16,131	9441	9059	2947	742	346	229	218
# of respondents to our survey	246	26	43	9	109	20	0	0	13
Sample size	0.62%	0.16%	0.46%	0.10%	3.6%	2.70%	0.00%	0.00%	5.96%

Data in the first two rows have been adapted from Statistics Canada (2010).

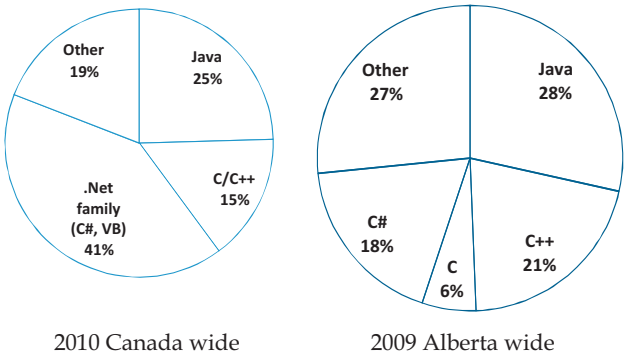

Fig. 14. Programming languages used for development.

larger in size compared to firms in other provinces. The average values in Fig. 13 somewhat relate to the lower end of the company size distribution in Fig. 12.

4.1.10. Programming languages used for development (Q11)

Similar to our 2009 Alberta-wide survey, the type of programming languages used was also gathered. The results are shown in Fig. 14. Most firms use the .Net family programming languages, followed by Java and C/C++. Other programming languages, including Erlang, Ruby, PHP, JavaScript, SQL, etc., are also reported, classified under "other" in the graph.

In order to compare the provincial and the national survey results, 2009 survey results are also shown in Fig. 14. Notable differences can be spotted. While .Net family covers the largest portion (41%) within the nation-wide range, most Albertan companies (28%) responded with Java as their programming language, yet C# only takes up 18%.

The choice of programming languages used for development can have important implications for the testing practices of a firm. For example, some programming languages (such as Java and C#) have strong automated test framework support (e.g., Junit and NUnit) and also industry standard development environments (such as the Eclipse and Microsoft Visual Studio).

4.2. Test types and levels

To study test types and levels, we measured the following data:

- Test types (Q12).
- Effort spent on each type of testing (Q13).
- Testing across the development life-cycle (Q14).

4.2.1. Test types (Q12)

In terms of frequency of usage of different test types, the survey results are presented in Fig. 15. We can see that unit and functional/system testing are the two most common test types. Other types including acceptance, GUI and performance testing also received a large portion of votes from the respondents. We found it surprising that some respondents reported that they do not conduct unit (50 respondents) or even system testing (54 respondents). In the "other" type, respondents mentioned other specific test types such as: infrastructure testing (testing of failure and recovery), and data transformation testing.

4.2.2. Effort spent on each type of testing (Q13)

In this question, the respondents were asked about how much emphasis in percentage they put on each test activity out of the total testing time and budget. For example, one respondent could mention that her team would spend 50% of the testing efforts on unit testing, 20% on system testing and 30% on performance testing. As

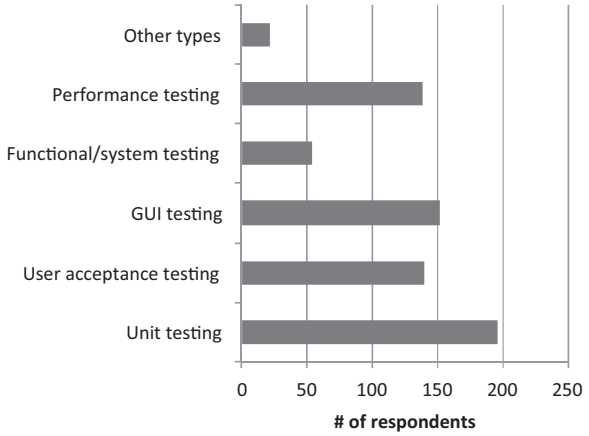

Fig. 15. Test types.

Fig. 16 shows, functional testing and unit testing received the most emphasis (excluding the "other" types). On the other hand, load and stress testing gained less emphasis, which reflected in its low feedback number and average emphasis rate. One possible indication is that what the managers emphasized actually influenced the test types that have been performed. The previous question (test types used), reported in Section 4.2.1, and this question somewhat complemented each other. The former measured what test types are used, and the latter (current) question asked the quantitative amount of emphasis (in percentage values) on each test type.

Also, it is interesting to observe different "variances" in the histograms for different testing types, e.g., different respondents have much wider emphasis on unit testing (between 0 and 100%) than load testing for example (between 0 and 15%). In the "other" type, respondents mentioned other specific test types such as: infrastructure testing (testing of failure and recovery), and data transformation testing.

4.2.3. Testing across the development life-cycle (Q14)

Participants were then asked about the type and phasing of their software development life-cycle processes and results are shown in Fig. 17. We can see that Test-last Development (TLD) approaches (i.e., testing after development) are still much more popular than Test-driven (first) Development (TDD), with a ratio close to 3:1 (146 respondents versus 50). A small portion of firms employ the Behavior-Driven Development (BDD) approach. The result indicates that traditional Test-last Development approach is still dominant among Canadian firms. The "other" category included types such as: iterative testing, and both TLD and TDD.

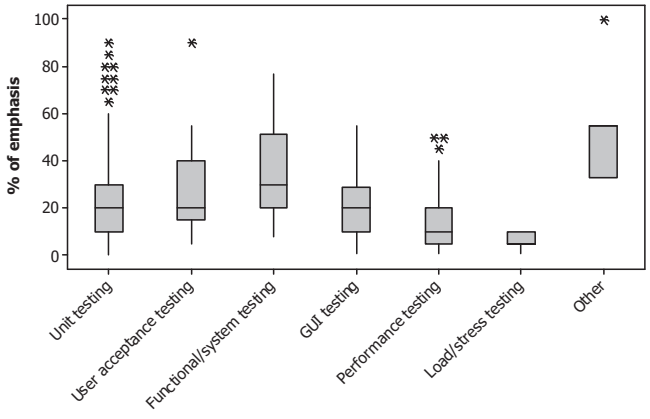

Fig. 16. Effort (emphasis) on various test types (in percentage).

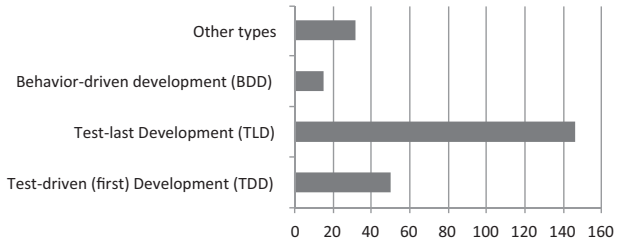

Fig. 17. Testing across the development life-cycle.

We will use this data in upcoming sections, e.g., in Section 4.5.1 to correlate the type of testing phase (TDD versus TLD) with the use of code coverage metrics, i.e., to assess whether coverage metrics are more popular in TDD or in TLD. Although code coverage metrics have been utilized for many years in traditional TLD approaches, the TDD is encouraging more wider use of coverage metrics and we are interested to evaluate this hypothesis.

4.3. Test techniques

To study the types of test techniques used, we measured the following data:

- Test-case generation techniques (Q15).
- Familiarity with and usage of mutation testing (Q16).

4.3.1. Test-case generation techniques (Q15)

Test techniques refer to the methods that software developers use for generating test cases. Results are shown in Fig. 18. It is surprising and unfortunate that most respondents do not use any explicit test-case generation technique (i.e., black-box or white-box techniques). Since approaches labeled 1, 2 and 3 are all black-box testing approaches, we can see that black-box testing approaches are more popular than white-box testing (based on code coverage). We can see that boundary-value analysis (Mathur, 2008) is the technique reported to be used by most respondents, followed by category partitioning. Among the approach reported in the "Other" category are: fuzzy (random) testing and user-story-based acceptance testing. On the positive side, it is good to see a good number of respondents using model-based testing (e.g., based on UML models).

Another surprising finding is the low popularity of exploratory testing. In the Agile practices, exploratory testing is widely recommended as an effective type of testing (Crispin and Gregory, 2008), but it seems that exploratory testing is not that popular among Canadian firms.

One possible factor that we tried to use to comprehend the difference in using various test-case generation techniques was to co-relate it to participants' years of work experience in software testing/QA. The individual-value plot in Fig. 19 visualizes that information. Again, we cannot observe a strong differentiating

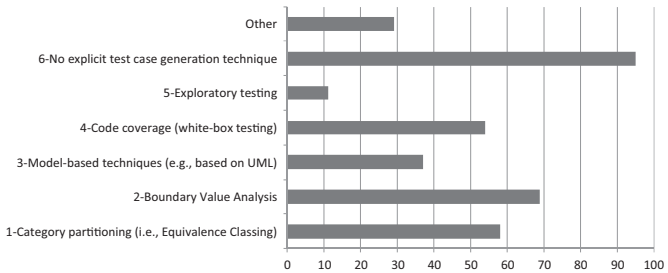

Fig. 18. Test-case generation techniques.

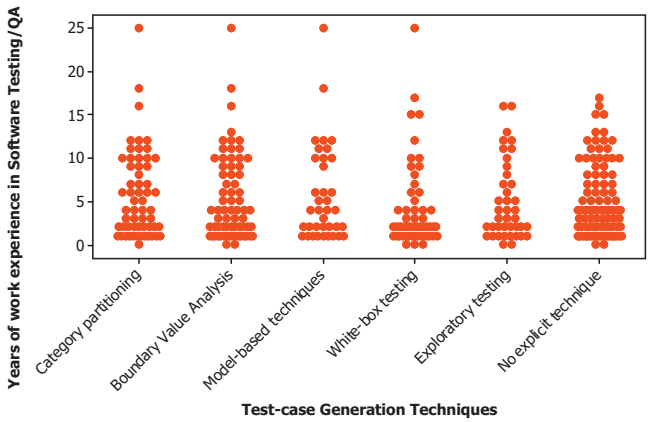

Fig. 19. Use of test-case generation techniques versus years of work experience in software testing/QA.

trend among different test-case generation techniques, as to reveal whether testers with more years of work experience would use different (more systematic or less systematic) test-case generation techniques. We can see that the distributions for "Exploratory testing" and "No explicit technique" have slightly less points in the higher spectrums of years of work experience, which is as one would expect. Thus, we see that, independent of number years of work experience, different participants are using different test-case generation techniques.

4.3.2. Familiarity with and usage of mutation testing (Q16)

Respondents were asked whether they were familiar with mutation testing (fault injection) and were using it. Fig. 20 shows that more than half of all respondents responded that they are they were not familiar with mutation testing. Yet about 70 respondents reported that they used mutation testing in some of their projects. Only one respondent reported using mutation testing in all of projects in his/her workplace.

As reported by Jia and Harman in a comprehensive survey of mutation testing (Jia and Harman, 2011), one possible reason for unfamiliarity of many practitioners about mutation testing could be due to shortage of industry-scale tool support.

4.4. Test automation and test tools

To study the extent of test automation and the types of test tools used, we measured the following data:

- Efforts on automated versus manual testing (Q17).
- Test tools and frameworks used (Q18).

4.4.1. Efforts on automated versus manual testing (Q17)

In this question, participants were asked to provide estimated percentages for automated and manual testing in their past projects. The summation of two numbers should be 100% for each participant. Results are shown as a box-plot and a line chart in

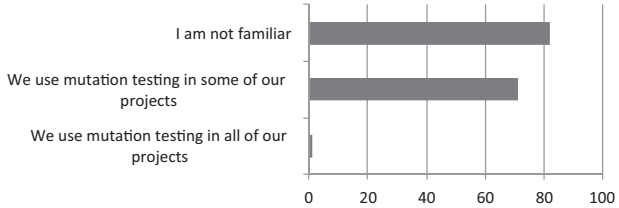

Fig. 20. Familiarity with mutation testing.

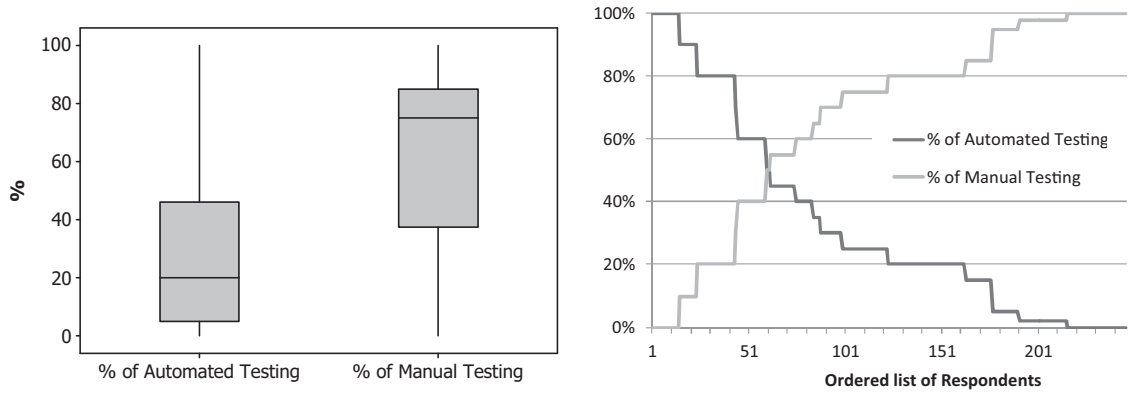

Fig. 21. Automated versus manual testing.

Fig. 21. Manual testing is taking the dominant position, given that nearly 100 respondents mentioned that they perform more than 80% of their testing manually. The usage of automated testing is not as popular as manual testing, as only43 respondents answered that 80% to 100% of their past testing practices have been automated testing. This indicates that test automation still has room for more adoption among the Canadian software companies.

To investigate potential factors which could correlate with percentage of automation for a given respondent, we hypothesized about three potential factors: (1) years of work experience in testing/QA, (2) company size, and (3) the phase of testing approaches across the development life-cycle (TDD versus TLD). We discuss next the correlation of each of these factors on percentage of test automation, based on the survey responses.

Fig. 22 shows, as an *X*–*Y* scatter plot, the correlation between years of work experience in testing/QA and percentage of automated testing. The Pearson correlation coefficient for the two variables is 0.04 (*p*-value = 0.54), thus not indicating any correlation.

Fig. 23 shows the percentage of automated testing used versus the company size values, as reported by respondents. We attempted to establish whether a higher percentage of automated testing is used in larger firms. As we can observe in Fig. 23, no major correlation could be established between these two factors.

Fig. 24 shows, as a box plot, the distribution of percentage values of automated testing and the type of testing approaches as reported by respondents. We excluded the responses which did not mention TDD or TLD. The two distributions somewhat provide interesting results. Except about 20 "outlier" points in the Test-last

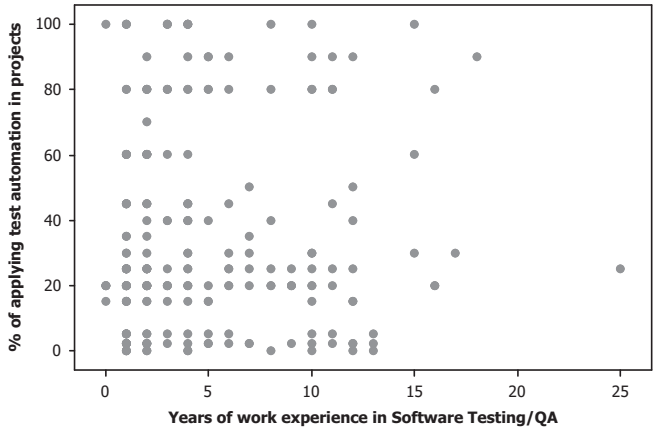

Fig. 22. Correlation between years of work experience in testing/QA and percentage of automated testing.

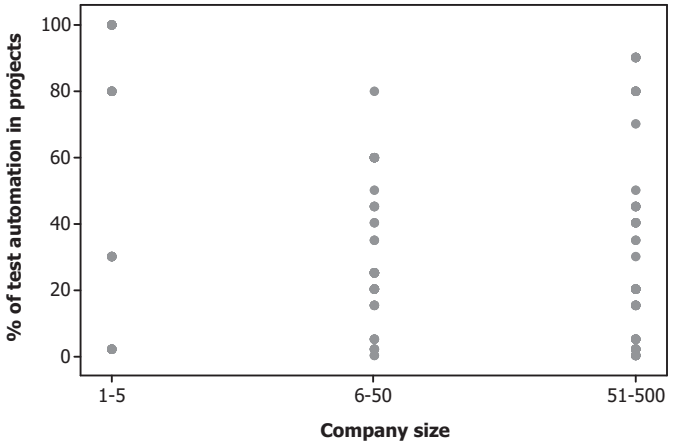

Fig. 23. Correlation between company size and percentage of automated testing.

Development (TLD) distribution (total of 146 points), most of the respondents who mentioned using TLD reported smaller percentage values of automated testing compared to adopters of TDD. This is while the major mass of the TDD distribution seems to be higher than the TLD distribution. Thus, while we cannot say with certainty that TDD users are using automated testing more, a weak trend is observable.

4.4.2. Test tools and frameworks used (Q18)

Inspired by the 2009 survey, we designed this question to know about the test tools and frameworks that Canadian firms are using. The categories provided were:

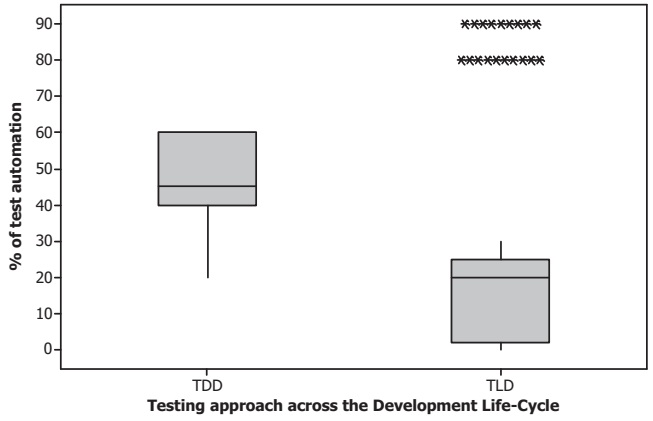

Fig. 24. Correlation between testing approach and percentage of automated testing.

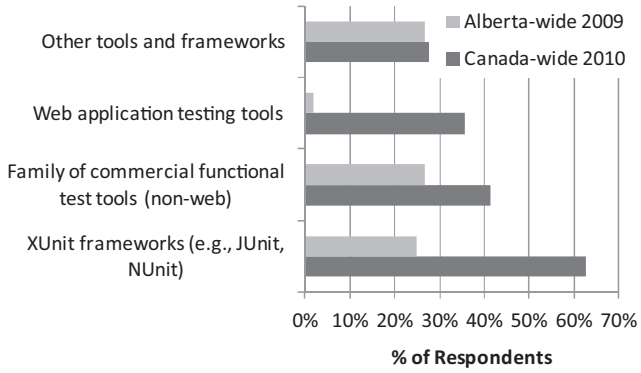

Fig. 25. Use of test tools and frameworks: comparison of the 2010 Canada-wide survey with the Alberta-wide results.

- XUnit frameworks, e.g., JUnit, NUnit.
- Family of commercial functional test tools (for testing non-web-based applications), e.g., IBM Rational Functional Tester.
- Web application testing tools, e.g., Selenium.
- Other tools and frameworks.

Fig. 25 shows the percentage of participants which indicated each of the above categories and compares the 2010 Canada-wide results with the 2009 Alberta-wide survey results. It turned out that XUnit frameworks and different commercial functional test tools are two of the most widely used test tools. Web application testing tools are also widely used.

If we compare the 2010 Canada-wide results with the 2009 Alberta-wide results, we find out that the Canada-wide percentage values are generally higher than the Alberta-wide percentage, indicating that it seems software engineers in other Canadian provinces are more expected to use test tools and frameworks compared to those in Alberta.

4.5. Test metrics

To study the usage of test-related metrics, we measured the following data:

- Code coverage metrics (Q19).
- Other test and quality metrics (Q20).

4.5.1. Code coverage metrics (Q19)

Measurement is instrumental to software testing and quality (Abran et al., 2001). Therefore it was important to us to determine the extent to which software organizations in Canada use measurements to plan their testing activities. As a form of white-box testing, code coverage can serve as a measurement of how much of the target software's source code has been tested. In this question, we provided four major coverage metrics and an "other" field. The four major coverage metrics were: line, decision, condition, and Modified Condition/Decision Coverage (MC/DC) (Mathur, 2008).

Question responses are shown in Fig. 26. 107 respondents (43% of the pool) mentioned they use no coverage metrics. From the bar-chart, we can see that decision (branch) coverage and condition coverage are two most popular coverage types among all the respondents. Line (statement) coverage also received more than 40 responses. The MC/DC is often used for the development of safety-critical software, gained the least votes. 40 respondents (16%) mentioned using "other" types of coverage metrics, such as: requirements coverage, and "whatever Microsoft tools provide".

We also wanted to know if the practitioners use the coverage metrics in combinations (more than one metric). The results are shown in Fig. 27. Over 70 respondents mentioned they use two

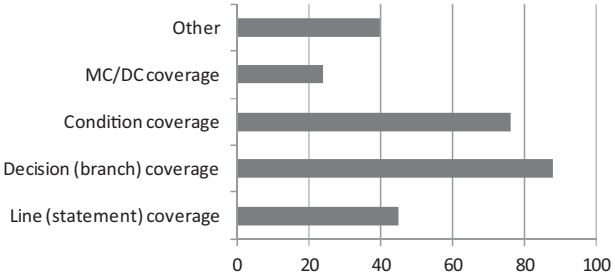

Fig. 26. Code (test) coverage metrics.

of the coverage metrics in their projects. No respondents reported using all the four criteria listed in our question.

Recall from Section 4.2.3 where the survey Q#14 asked participants about their use of testing across the development life-cycle (i.e., TDD versus TLD). Although code coverage metrics have been utilized for many years in traditional TLD approaches, the TDD is encouraging more wider use of coverage metrics and we are interested to evaluate the hypothesis that whether coverage metrics are more popular in TDD than in TLD. To analyze this, we correlated each respondent's response for both questions and calculated the number of times the usage of a type of coverage metric was reported for TDD and also for TLD. The results are as follows: out of 50 respondents who reported usage of TDD, 37 reported using coverage metrics (37/50, 74%). On the other hand, out of 146 respondents who reported usage of TLD, 83 reported using coverage metrics (83/146, 56%). Thus, it seems coverage metrics are somewhat more popular in TDD than in TLD.

We also hypothesized that testers with more years of work experience would use more types of coverage metrics (Fig. 27). We calculated the correlation between the years of work experience for each tester participant and the number of coverage metrics he/she reported using (0 for no metrics). The Pearson correlation coefficient for these two data series was −0.04 (very close to zero) which showed no support for the above hypothesis, i.e., testers with more years of work experience do not necessarily use coverage metrics or more types of them.

4.5.2. Other test and quality metrics (Q20)

In addition to code coverage metrics, participants were also asked to select from a given list of other test and quality-related metrics that they used explicitly in their past and current projects (Fig. 28). Among all the listed metrics, number of passing user acceptance tests attracted the most votes, followed by total number of defect detected per day (week, or month). The results from this question indicate the perceived significance of user acceptance and defect detection rate in testing activities in Canadian firms. Other metrics, including number of test cases executed within a period of time and defect per line of code, also received a reasonable number of votes.

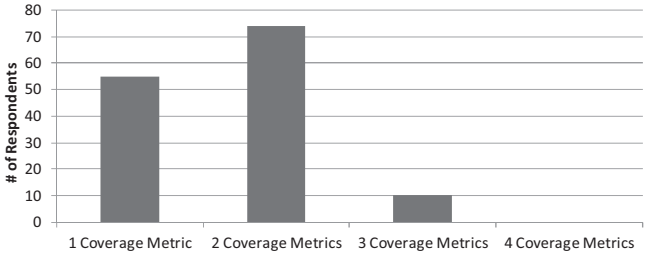

Fig. 27. Using coverage metrics in combinations.

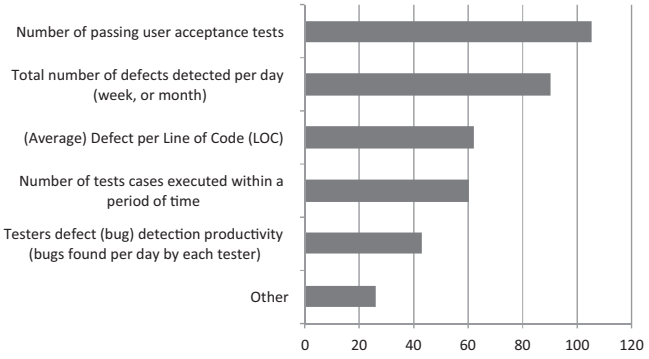

Fig. 28. Other test and quality metrics.

Similar to Q19, we also hypothesized that testers with more years of work experience would use more types of the above five other metrics (as shown in Fig. 28). We calculated the correlation between the years of work experience for each tester participant and the number of the above five other metrics he/she reported using (0 for no metrics). The Pearson correlation coefficient for these two data series was 0.31 indicating a weak correlation in support of the above hypothesis, i.e., testers with more years of work experience did use more of the above five other metrics to some extent.

4.6. Test management

To study test-management issues, we measured the following data:

- Ratio of testers to developers (Q21)
- Criteria for terminating testing phase (Q22)
- Barriers for adoption of testing methodology and tools (Q23)
- Effort spent on testing during the software development process (Q24)

4.6.1. Ratio of testers to developers (Q21)

This question was designed to find out the tester to developer ratio used in different teams. Results are shown in Fig. 29. In most companies, testers are outnumbered by developers, with ratios ranging from 1:2 to 1:5. According to the figure, the frequency is decreasing as the (tester:developer) proportion grows from 1:5 to 1:2. Besides, a substantial amount of companies make no distinction between these two roles.

In fact, the tester to developer ratio (tester:developer) has been an actively debated issue in the software industry lately. James Whittaker, a practitioner test engineering manager has a blog post (Whittaker, 2008) on the topic where he compares two leading software companies, in terms for their tester:developer ratio. He points out that in the two selected large leading technology companies, the ratio varies between 1:1 and 1:3. He finally concludes that: "*Test managers should be trying to find that sweet spot*" (Whittaker, 2008). Iberle and Bartlett have an online article (Iberle and Bartlett, 2001) on how to tackle this problem for a company, i.e., estimating the suitable tester to developer ratio. They present an interesting model and approach to come up with realistic tester to developer ratio for any given company/project. In our results, it seems that the most popular "sweet spot" chosen by managers have been from 1:2 to 1:5.

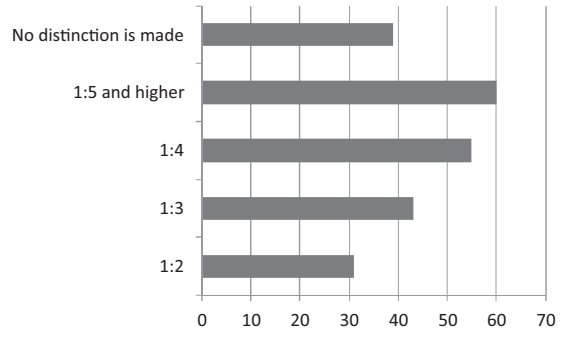

Fig. 29. Ratio of testers to developers (number of testers:number of developers).

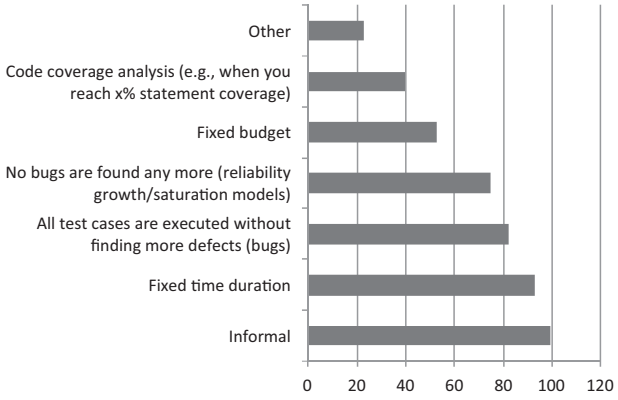

Fig. 30. Criteria for terminating testing activity.

There are other studies on the issue of tester to developer ratio. Cusumano and Yoffie (1999) report about the competition of Microsoft and Netscape in 1998 in browser development. They report that Netscape employed fewer testers and they then examine the implications of such a decision to other aspects of the development process (e.g., release time).

4.6.2. Criteria for terminating testing phase (Q22)

The criteria that firms use to decide when to stop testing remain a major problem for most software organizations (Mathur, 2008). The problem lies in determining when the quality of the product is "good enough" for the organization to consider releasing it. The economics of testing and time to market are interconnected and come with a trade-off, making this decision a complex and risky one. Statistics of this issue from our survey results is shown in Fig. 30. Apparently, most companies do not formally use any criteria (labeled as "informal").

Apart from that, a number of companies will set duration for testing phase. When the fixed time ends, testing terminates. On the other hand, the number of found bugs and reliability growth are two popular criteria for some companies. Besides, some respondents chose fixed budget as their criteria. Compared with the mentioned criteria, code coverage, which is a technical metric for testing itself, received less attention. Nevertheless, the percentage of using coverage analysis as termination criteria is interestingly higher in this Canada-wide survey (40%) than that in 2009 Alberta-wide survey (29%), indicating that Albertan firms might fall behind the national average level in adopting technical evidence as test termination criteria.

4.6.3. Barriers for adoption of testing methodology and tools (Q23)

The respondents were also asked about the barriers for adoption of testing methodology and tools in their companies. Not surprisingly, cost, time and lack of support from high-level management were the three top responses for this question. Results are shown in Fig. 31 (the *X*-axis is in logarithmic scale). We can see that cost is ranked the number one barrier in both the 2009 and 2010 surveys.

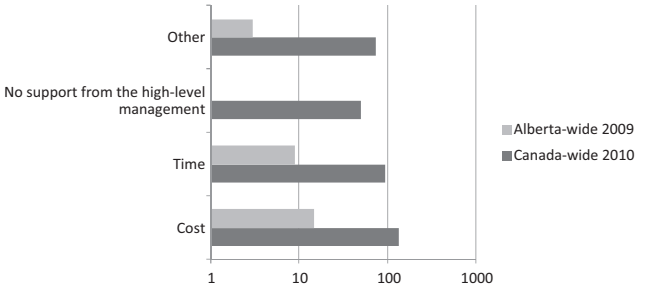

Fig. 31. Barriers for adoption of testing methodology and tools.

It is interesting to see that lack of support from high-level management has also been a major reason. Lack of time for adapting testing methodologies and tools might be due to tight schedule or lack of comprehensive planning, which can be again mostly a managerial concern. Among the frequent other reasons were: organization not mature enough, and no apparent need for testing methodology and tools.

While searching for "related work" for this aspect of our survey, we found an excellent keynote paper by Cordy (2003) who explored the practical barriers to industrial adoption of software maintenance automation techniques. He discussed some of the social, technical and business realities that lie at the root of this resistance. Cordy reported similar types of barriers in the software maintenance domain as well.

As one possible solution to these barriers, we have found that the testing practitioners should be "convinced" that systematic testing methodologies are not that costly after all, if they are planned and conducted carefully (Jolly et al., 2012; Ali et al., 2012; Pinheiro et al., 2010). Along with other researchers, we have been quite successful in applying the "action research" approach and conducting technology transfer of systematic testing methodologies to real-world projects (Jolly et al., 2012; Ali et al., 2012; Pinheiro et al., 2010), and in return, have observed that the above so-called barriers are slowly fading away.

4.6.4. Effort spent on testing during the software development process (Q24)

Anecdotal evidence from software developers suggests that testing is becoming an increasing percentage of the development budget (Gelperin and Hetzel, 1988). To find out the share of effort spent on testing during the software development process in Canadian firms, respondents were asked to estimate a rough percentage number for how much budget and time they spent on testing. The box-plot of the data, and also the data histogram by mapping the responses into five equal internals between 0 and 100% are shown in Fig. 32. The average value is 28%.

The results show that only a small portion of respondents reported that they allocated more than 40% budget and time on testing, while the majority spent less than 40%. 6 out of 246

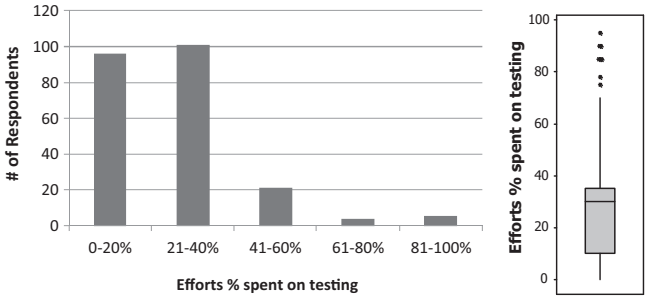

Fig. 32. Effort spent on testing.

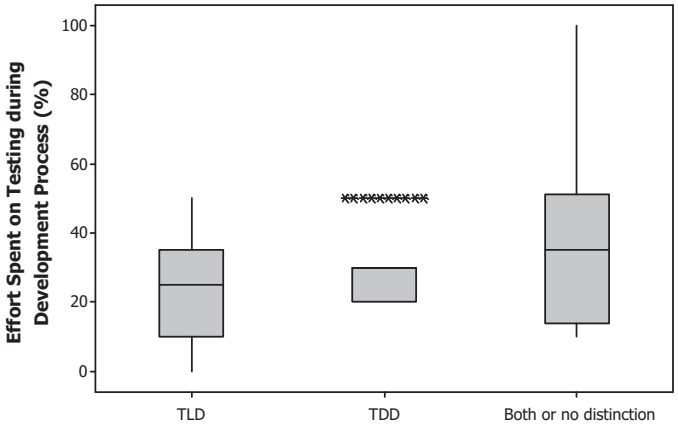

Fig. 33. Effort spent on testing versus testing approach across the development life-cycle (TDD versus TLD).

respondents mentioned they spend more than 80% of their development effort on testing. A survey conducted in Australia in 2004 showed a similar trend (Gelperin and Hetzel, 1988).

In terms of effort spent on testing during the software development process, we hypothesized that the effort values might be different for the two testing approaches across the development life-cycle (TDD versus TLD). We thus separated the effort percentage values, in the data set, for the respondents who reported TDD, TLD, and those who reported both or did not distinguish them. The three box plots are shown in Fig. 33. As we can see, there is no major difference (other than the variance) between the TDD and TLD distributions. In TLD, it is interesting to observe that, the lower end of the distribution is quite close to zero, meaning that in some projects, probably due to being rushed, the testing phase was bypassed altogether. Of course, a detailed follow-up study and analysis of the quality of such software systems after release would be interesting.

4.7. Test education and training

To study test training, we measured the following data:

- Formal training on software testing (Q25).
- Training programs (Q26).
- Barriers of software testing training (Q27).

4.7.1. Formal training on software testing (Q25)

Receiving formal training on software testing would usually provide a solid foundation for testers. In this section, training programs include university or on-site training courses, but exclude self-study hours based on online resources or books (this phrase was explicitly mentioned in the online survey). Responses were counted by hours and results are shown in Fig. 34. It is encouraging to see that more than half of 206 respondents received at least 20 h of formal training in a year on testing. This indicated the awareness of training importance among substantial number of Canadian firms. However, the rest of respondents (39%) mentioned that they received no training on testing in the year before the survey. The first author is actually active in offering corporate training classes for practitioner testers and has published a few papers in the area of software testing education research (e.g., Garousi, 2011a).

4.7.2. Training programs (Q26)

As Fig. 35 shows, in terms of training programs received by the respondents, functional/system testing and unit testing are ranked the two most popular types. The other popular but less common training types are performance testing and user acceptance testing.

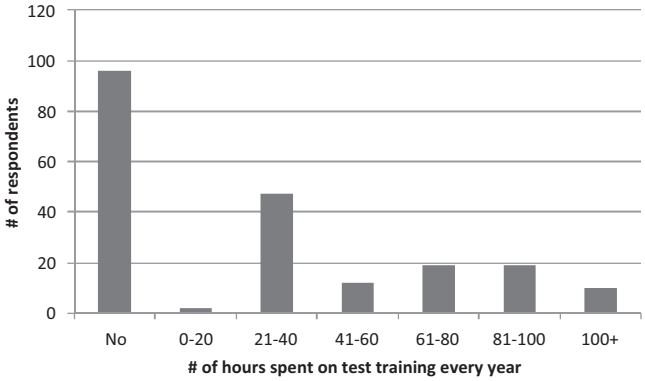

Fig. 34. Hours of formal training on software testing.

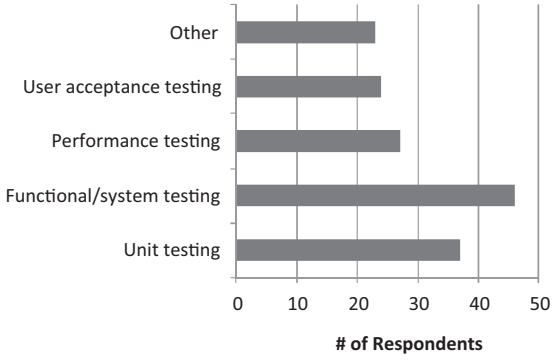

Fig. 35. Training programs.

It is not difficult to notice that this distribution correlates with testing types and emphasis survey results which are shown in Sections 4.2.1 and 4.2.2.

4.7.3. Barriers of software testing training (Q27)

To find out the reasons that prevent Canadian companies from providing software training to testing staff, responses to this question were collected and results are shown in Fig. 36. Interestingly, the same trend as Section 4.6.3(barriers for adoption of testing methodology and tools) occurred here. Cost is still ranked the first barrier, followed by time and lack of support from high-level management. Similar to Section 4.6.3, the barriers were either limited resources or managerial concerns.

4.8. Software testing research and interaction with university researchers

Under this category, we measured the following data:

- Involvement in software testing research (Q28).

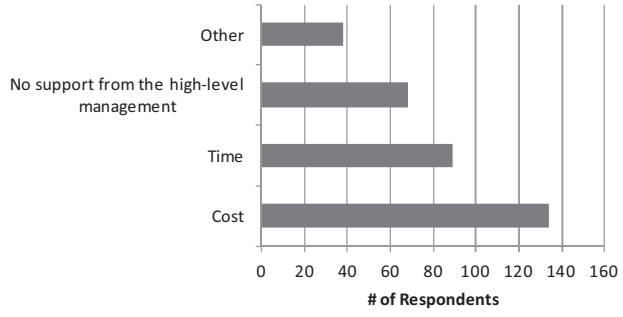

Fig. 36. Barriers of software testing training.

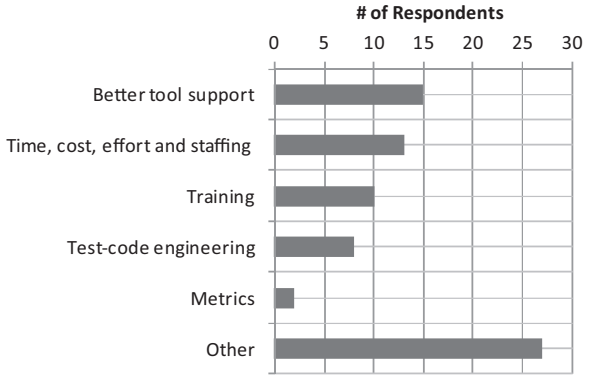

Fig. 37. Summary (classification) of challenges posed by survey respondents for the research community.

- Top challenges posed for research community (Q29).
- Frequency of interaction with research community (Q30).
- Attending testing conferences (Q31).
- Frequency of attending testing conferences (Q32).
- Frequency of reading technical testing papers (Q33).
- Frequency of participation in online discussion forums related to testing (Q34).

4.8.1. Involvement in software testing research (Q28)

Respondents were asked whether their companies or themselves involved in any research to develop new test techniques/tools. Surprisingly, 243 (98%) respondents reported "No" to this question while only 3 (2%) respondents replied "Yes". The handful number of respondents reported the following research endeavors: (1) automated exploratory testing, (2) speech recognition testing, and (3) developing an internal automated testing tool.

4.8.2. Challenges posed for research community (Q29)

In this question, respondents were asked to list the testing challenges that they have been seeing in their projects and would like research community to work on. The question was a free text field (i.e., a qualitative input instead of quantitative input). We were hoping that answers to this question would enable us to point out new research directions for the software testing research community.

From the pool of respondents, only 64 concrete responses were provided. As one would expect, we received a variety of qualitative responses, ranging from testing metrics to managerial concerns. We synthesized, aggregated and clustered the responses. The resulting classification of the responses in six categories is shown in Fig. 37. The raw (full) list of responses and also their detailed classification are provided in Appendix A.

15 respondents raised the need for better testing tools. From their point of view, new tools should be simple, highly customizable and capable of automating most of the testing tasks. 13 responses were related to time, cost, effort and staffing of testing tasks. For example, two of the reported challenges were: "Determining the value of testing versus its cost", and "No dedicated testers on most projects".

Challenges related to training (i.e., low expertise of testing staff) were the third most reported (10 responses). Two of the reported challenges in this category were: "Lack of trained staff for automation testing" and "Commercial tools are expensive – more promotion and education on adapting open source tools". Thus, it seems there is a need for systematic education and training of test engineers, and studies in the area of software testing education research (e.g., Garousi, 2011a, 2010a; Garousi and Mathur, 2010) are needed in this regard.

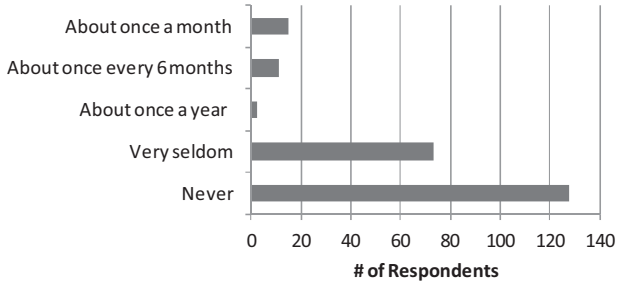

Fig. 38. Frequency of interaction with research community.

Eight responses were related to development and maintenance of test-code and issues related to the entire test-code life-cycle. We refer to the topic encompassing all those issues as *test-code engineering* as shown in Fig. 37. Two of the reported challenges in this category were: "Maintainability of the automated test suites", and "Reusability of test scripts would be beneficial".

Two responses were related to test metrics. They were: "Better techniques/metrics for deciding what should be tested", and "Reporting meaningful and useful test metrics". 27 responses were related to other types of issues, e.g.:

- Complexity of maintaining and documenting test plans and test results.
- Context-driven testing.
- Ease of integrating testing into development activities.
- Improper test planning.

4.8.3. Frequency of interaction with research community (Q30)

We wanted to know how often respondents interacted with academic software testing researchers about their test-related problems/challenges. The result is presented in Fig. 38. Disappointingly, the majority of respondents never (56%) or seldom (32%) interact with the researchers in academia. Those who interacted with researchers once a year or more only covered a small portion among all respondents (12%). This indicates that in the Canadian context, the interaction between research and industrial communities needs to be improved.

Effective communication between these two distinct communities can help researchers develop test techniques that would better suit the needs of industry. There are events such as the IBM CASCON (Centre for Advanced Studies Conference) (IBM, 2012) and the Workshop on Enhancing Software Engineering Practices among Alberta's Industry, Government, and Universities in Alberta (SEAB) (SERG, 2012), also consortiums such as the (Canadian) Consortium for Software Engineering (CSER) to facilitate this. However, it seems that more has to be done in this regard.

4.8.4. Attending testing conferences (Q31)

Attending testing conferences can help practitioners better understand the latest research findings and let these findings benefit the software industry. We listed a few widely known conference option in this question. However, most respondents mentioned that their companies did not provide funding for them to attend conferences. It is encouraging that in some conferences, such as STAREast or STARWest, ICST, and CAST, we can see some presence by industrial respondents.

STAREast or STARWest are two major industrial software testing conferences. ISSTA and ICST are two academic conferences. As per the first author's experience from attending both industrial and academic software testing conferences, there is a major disjoint (community segmentation) among the industrial and academic conferences, i.e., most academics do not attend industrial conferences, and, most practitioners do not attend academic conferences, since both sides do not find much value in those venues. There is a need for further research on this issue (Fig. 39).

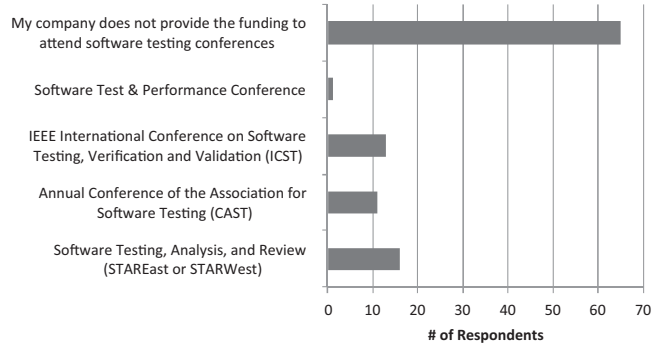

Fig. 39. Attending testing conferences.

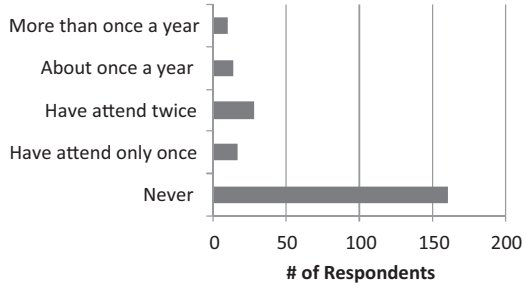

Fig. 40. Frequency of attending testing conferences.

4.8.5. Frequency of attending testing conferences (Q32)

We also asked the respondents, if attending conferences, how often they attend testing conferences. Results are shown in Fig. 40. The answer "Never" is dominant (70%). 7% and 12% of respondents mentioned that they have attended conference at least once or twice in the past, respectively. 6% and 4% mentioned that they attend on average once and more than once a year, respectively. The chart reflects a similar trend as Section 4.8.3: industrial practitioners lack substantial in-person interaction with the research community.

4.8.6. Frequency of reading technical testing papers (Q33)

Compared with attending conferences, reading testing papers/articles requires less time and money. In this question, respondents were asked about their reading frequency. Some representative magazines such as the IEEE Software were given. The result is rather encouraging (Fig. 41). The majority (45%) reported to read at least a paper/article once a month. There are a large portion of respondents who read only once every 6 months (19%) or once a year (4%). Still, some respondents very seldom (%21) or never (11%) read testing papers. We can imagine reading as a common means that practitioners can (should) use to

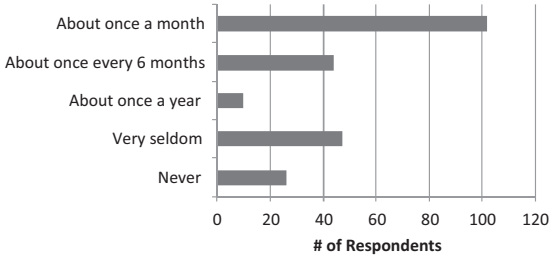

Fig. 41. Frequency of reading testing papers.

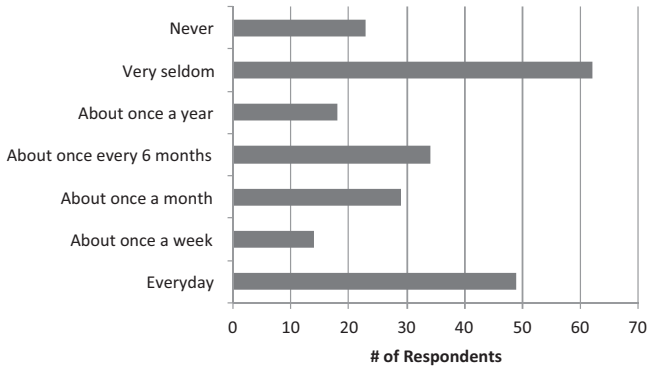

Fig. 42. Frequency of participation in online discussion forum.

update their testing-domain knowledge and adapt advanced test techniques in their projects.

4.8.7. Frequency of participation in online discussion forums related to testing (Q34)

Online tools such as mailing lists, blogs, newsletters or discussion forums (e.g., StackOverflow.com) are popular methods for practitioners to discuss technical problems and to learn new concepts. The more frequently practitioners participate in such forums, the more likely they are to keep their knowledge up-to-date. Encouragingly, a substantial number of respondents (21%) reported to participate in online forums related to testing every day. Also, some read/participated in online discussions in a weekly (6%) or monthly basis (13%). Similar to Section 4.8.6, 85 out of 250 respondents (37%) very seldom or never joined online discussions (Fig. 42).

5. Discussions

Summary of our findings are discussed in Section 5.1. Section 5.2 discusses the lessons learned. Potential threats to the validity of our study and steps we have taken to minimize or mitigate them are discussed in Section 5.3.

5.1. Summary of findings

Our survey gatheredresponsesfrom246 practitioner software engineers working in Canadian companies. According to our demographic data, software firms from all Canadian provinces except Prince Edward Island, New Brunswick, and Newfoundland and Labrador had responded to our survey. Organization types varied from those developing customized software to software development for internal use, and from software contractors to developers of packaged software. There was a good mix of different project and company profiles (Section 4.1) in our survey data, and this helped our results to be unbiased from certain types of development firms.

Among other results, the survey revealed some different trends from the 2009 Alberta-wide survey. One difference lies in the test training aspect (Q25–Q27, Section 4.7). While 2009 survey results revealed the substantial need for adequate training, in current survey results we witnessed some encouraging results, namely, that more than half of 206 respondents reported to have received at least 20 h of test training in a year (Q25). The awareness of importance of training is an indicator of awareness for product quality and that managers are sponsoring employees to acquire and improve their testing skills for the goal of creating high-quality products.

Although the trend about testing training is encouraging, several other findings show areas of concern. One of the negative findings is related to test termination criteria (Q22). The trend in Canada is similar to that in the 2009 Alberta-wide survey in that most firms do not use formal stopping criteria for testing. The popular approach used by Canadian firms to determine whether the test team should stop testing the product is to base the decision on managerial concerns, for example, setting fixed time/budget for testing, or to decide based on primary bug-detection results, for instance, no bugs found for a period of time. More systematic approaches, such as code coverage analysis, received relatively less attention. This is probably due to the fact that testers or managers lack formalized test-related training which leads to their lack of understanding of or confidence in advanced test termination criteria. Lack of formal stopping criteria for testing increases the risk of releasing software products with defects. One other contributing factor is that most projects are under pressure due to limited budget and time (Section 4.6). These two resources are also considered as major barriers for adopting new testing methodology and tools (Q23) and for the testers to receive further training (Q27).

Similar to the 2009 Alberta survey, test-case generation still relies heavily on tester knowledge and intuition (Q15). This leads to the question of whether the testers' knowledge and intuition is a reliable source to derive test cases. Without further studies, the best answer seems to be that they were developers at an earlier stage of their careers, or they acquired their knowledge on the job. This is effective, but it is also probably the most expensive way to acquire testing knowledge.

Overall, some aspects of testing (e.g., training) were shown to be in a different trend, yet problems (e.g., termination criteria, automation, etc.) continue to exist by comparing the 2009 Alberta provincial survey and this Canada-wide one. It is believed that Canadian firms should reconsider the tradeoff between the risk of over-budget or delivery delay and training employees in testing skills, so as to achieve higher product quality. The ability to detect tendencies that lead to reduced quality and to identify the root causes of reductions in product quality may suffer from the lack of testing. The 2002 National Institute of Standards and Technology (NIST) report suggests that the net negative economic impact of a lack of testing infrastructure in the U.S. amounts to 6.2 billion US dollars per year (Tassey, 2002). The relative impact in Canada is probably similar, and therefore there is a need for increased quality assurance vigilance.

By comparing the current Canadian survey with some other international studies conducted in other countries or regions (e.g., US, Sweden, and Australia), we had some interesting findings. What we found common among the studies are:

- Manual testing is still in the dominant position versus automated testing (Section 4.4). This trend is similar in the Australia survey reported in Taipale et al. (2005).
- Black-box testing approaches are more popular than white-box testing, which is similar to the Australian survey results (Taipale et al., 2005).

It is also interesting to find out what is unique in Canada in terms of testing practices. Note that, since the questions asked in other international surveys were mostly different from ours, we can only compare a small portion of the findings. Other results are considered unique in this survey because those questions have not appeared in the other related surveys. It indicates probably, but not necessarily, that the described situation is unique in Canada. Those findings include:

- Functional and unit testing are two common test types that receive the most attention and efforts, followed by GUI, acceptance and performance testing (Section 4.2).

- Compared with the Australian survey (Taipale et al., 2005), the usage of mutation testing is getting some minimal attention in Canadian firms (Section 4.3.2).
- Traditional Test-last Development (TLD) style is taking the dominant position. Few companies are attempting the new development approaches (e.g., TDD, BDD, etc.), as reported in Section 4.2.3.
- NUnit and Web application testing tools are two most popular test tools used in Canadian firms, surprisingly overtaking JUnit and IBM Rational tools (Section 4.4).
- Most Canadian companies used a combination of two coverage metrics in their projects: decision (branch) and condition coverage are more popular than other metrics (Section 4.5).
- Number of passing user acceptance tests and number of defects found per day (week or month) are regarded as two important metrics for software quality (Section 4.6).
- In most Canadian companies, testers are outnumbered by developers, with ratios ranging from 1:2 to 1:5 (Section 4.6.1).
- The majority of Canadian firms spent less than 40% efforts (budget and time) on testing during development process.
- The gap between testing industry and research remains huge (Section 4.8). This is indicated by extremely low research involvement rate, low frequency of interacting with research community and of attending testing conferences.
- More than 70% of respondents participated in online discussion forum (Section 4.8.7), which indicated an encouraging trend that testers are actively communicating and learning from each other.

It is the authors' belief that general improvement in various testing practice factors from 2004 to 2009 in Alberta and also in Canada in 2010 should have been partially due to (1) world-wide increased awareness and appreciation of the importance of systematic software testing, (2) possibly due to the results of the 2004 and 2009 surveys, and (3) also a series of several joint industry–academia software testing workshops held by Canadian secondary educational institutes, e.g., in the University of Calgary, since 2005.

Last but not least, from the answers to Q29 (top challenges posed for research community) and other relevant questions, a number of research questions are raised for the research community:

- What is the rate of increase in size and complexity of software systems versus the reaction of the software industry to spend more efforts on testing them?
- What are the implications of the tester to developer ratio (if any) on the overall quality of the software developed by a team or organization?
- In terms of the informal criteria used by companies for terminating the testing phase, it would be beneficial to empirically evaluate the usefulness of different testing termination criteria in different projects.
- What are the suitable approaches to decide what should be tested and what should not? What is the tradeoff between time spent on testing and the achieved product quality? In other words, we should seek the most economically efficient way to perform software testing.

5.2. Lessons learned

One lesson learned from conducting this survey is that the survey strategy is important. Our survey is a replication of our earlier two survey studies (Geras et al., 2004; Garousi and Varma, 2010). The success of these two former surveys was helpful for conducting the current survey both in questionnaire design and execution. As mentioned Section 3.1, the questions in 2004 and 2009 studies were designed using SWEBOK categorization of test knowledge. From the experience of analyzing the results of two studies, we designed the questionnaire in a similar structure for this survey and elaborated on some questions, based on the lessons learned from the earlier surveys, making it more capable of revealing the software testing practices. During the execution phase, the online survey system provided a paperless, easily accessible questionnaire that was of great help in ensuring that we got good response samples from all provinces across Canada.

Challenges we had are similar to (Cater-Steel, 2005), successful testing practice should be seen related to economics and psychology issues rather than the result of purely technical considerations. Whether it is the problem of adopting which type of testing methodology or tools, to decide extent to which testing is enough, to adopt fine-grained coverage criteria, or to provide training for employees, those questions finally relate to balancing resources and cost. It is believed that Canadian firms are aware of these software testing practice issues. However, other mundane, organizational concerns such as timing of releases, competition pressure, etc., might be put into a higher priority than technical concerns.

Not surprisingly, the disconnect between research and industry, revealed by the survey results, is worthy of careful investigation by testing researchers, at least in Canada. Motivated by the need to further expand collaborations with the industry by our survey results, the authors have started to become more involved with the industry, for example by invited talks (Garousi, 2011b, 2010b) and technology-transfer workshops for practitioners (Garousi, 2010c). Such initiatives are needed from both sides (academia and industry) to tackle the key research challenges in this area.

We also received valuable feedback to improve several of the questions for future surveys, e.g.:

- For Q8 (Agile versus traditional development methodologies), we learned that it would have been nice to also have another option for this question such as "both Agile and traditional" or "depending on the project type".
- For Q31 (attending testing conferences), we learned that, given that the answers came from varying groups of people, it cannot be fairly expected that the participants have personally participated a conference. More so, if they did not, but their company did, they may not be expected to be able to recall the name of the conference. Considering this, the better option could be to classify the answers as:
  - I have attended.
  - My team has attended.
  - Our company has attended.
  - We do not attend.
  - I do not know.

5.3. Threats to validity

One threat to external validity lies in the demographic distribution of response samples. As reported in Section 3.2, our participants were mainly invited through the researchers' network of partners/contracts in Canadian software companies. Companies out of our contact were not probably properly represented in the survey population. As a result, responses from Alberta where the researchers were located were relatively higher than the other provinces. This probably introduced biases of results since the characteristics of Albertan firms could be different from those in other provinces. To mitigate this validity issue, we attempted to enlarge our email-list to include all major Canadian companies and preserved three months for all invitees to finish the online questionnaire. Fortunately, our survey received responses from 246 out of estimated 11,754 Canadian testing practitioners (sample rate: 246/11,754 = 2.09%) whose locations varied and covered nearly all major areas in Canada.

In terms of construct validity, the issue relates to whether we are actually measuring the real-world test practice in this survey. What we did was common to other survey studies – we counted the votes for each question and then made statistical inferences. It is believed that results based on such voting data can, to a certain extent, reflect the opinions of the majority of Canadian practitioners.

On the other hand, the validity of our conclusions could be of our concern. We attempted to conclude, qualitatively, that the problem of testing practice is more of economics and psychology issues rather than purely technical concerns. This is a statement inspired by (Jia and Harman, 2011). For our survey, we attempted to reduce the bias by seeking support from the statistical results of each testing aspect (test type, technique, automation, etc.).

Finally, we did not intend to establish any causal relationship in this study, thus the discussion of internal validity is omitted in this section.

6. Conclusions and future works

Thanks to the results of this survey, the authors have observed that the Canadian industry is paying more attention to the importance of testing lately. This can be observed by rising awareness of testing-related training. Also, the first author has been providing industrial training classes and events on testing in Alberta in the last several years and he has observed the increased popularity of the classes. Wider acceptance of the Agile methods in which testing plays a major role seem to also have contributed significantly to the increased popularity of software testing and its training in Alberta and in Canada.

Further similar studies are needed in other regions and countries to be able to compare the latest trends in software testing practices. It would also be a good idea to assess the maturity of testing practices in Alberta and elsewhere in future works by aligning them to standards such as the *Test Maturity Model Integration (TMMI)* (TMMi Foundation, 2010).

Also, the participants in our study were from several different roles (e.g., developers and managers). It would have been possible to analyze the results per each role group, but for space constraints, we analyzed the results all in one group. For future work, the survey group should be more focused, either toward company-level, organization-level or project-level participants.

Recall from results of Q18 that we asked the participants the exact names of test tools and frameworks that they mostly use. In addition to (or even, instead of) that question, we plan to include in our future surveys a question on the intended usage of test tools (e.g., case manager, automation tool, GUI tester, stress tester etc.) instead of tool (vendor) names.

Acknowledgements

Vahid Garousi was supported by the Discovery Grant no. 341511-07 from the Natural Sciences and Engineering Research Council of Canada (NSERC), by the Visiting Scientist Fellowship Program (#2221) of the Scientific and Technological Research Council of Turkey (TUBITAK), and also by the Alberta Ingenuity New Faculty Award no. 200600673.

The authors would like to thank Frank Maurer, Shelly Park, Christian Wiederseiner, Ayse Bener, and Ayse Tosun for their help in publicity for the survey and inviting their industry contacts to participate in this survey. We also would like to thank Janet Gregory for her feedback on earlier versions of the questionnaire. The authors would also like to thank the anonymous survey participants for their time and participations. Last but not least, the authors would like to thank the anonymous reviewers for their insightful comments in the revision process of this article.

Appendix A. Challenges raised by survey respondents for the research community

Challenges	Categories					
	Better tool support	Time, effort and staffing related	Training	Test-code engineering	Metrics	Other
Automated GUI testing – not supper stable	x					
Automated test script generation based on functional specifications	x					
Automation scope	x					
Better techniques/metrics for deciding what should be tested					x	
Budget and resource needs around test automation		x				
Capture testing steps and reproducing the same conditions reliably	x			x		
Commercial tools are expensive – more promotion and education on adapting open source tools	x		x			
Complexity of maintaining and documenting test plans and test results						x
Confusion among QA, QC, Testing, Validation and Verification						x
Context Driven Testing						x
Critical thinking – testers are not monkeys, we have critical thought, training needed!			x			x
Data generator tools	x					
Deployment failures						x
Determining the value of testing versus cost of testing		x				
Difficulty in estimating time/effort required to undertake rigorous testing		x				
Difficulty to estimate test overlap (leading to useless redundant testing)						x
Ease of integrating testing into development activities						x
Easy automation of unit tests into larger system test suites	x			x		
Exploratory testing/Session Based						x
Fidelity of testing when compared to actual, real world use (e.g., it passed all the tests, but *still* failed in the field)						x
Focus on web-based software, with automatic test case creation	x			x		
GUI testing tools to support C++, Delphi and Fortran are difficult to find/use	x					
Improper planning						x
Infrastructure testing						x
Insufficient time		x				
Insufficient time allowed		x				
Insufficient understanding by senior management						x

Challenges	Categories					
	Better tool support	Time, effort and staffing related	Training	Test-code engineering	Metrics	Other
Integration of build, install, test process						x
Integration testing						x
Lack of developer commitment, belief, and training around QA		x	x			
Lack of trained staff for automation testing		x	x			
Lack of training			x			
Less workers more work as the project progresses		x				
Maintainability of the automated test suites				x		x
More structured testing process						x
Most of the present day testing techniques are time consuming		x				
Need simple, easy to configure tools that automate much of the testing tasks (provide very rich out-of-the-box experience)	x					
No dedicated testers on most projects		x				
No requirements						x
No training			x			
Non testable legacy code				x		
Not enough exposure			x			
Organization not mature enough						x
Overly rigid automated tests make code difficult to refactor				x		x
Poor tools	x					
Quantitatively demonstrating the value of testing						x
Reporting meaningful and useful test metrics					x	
Result analysis tools	x					
Reusability of test scripts would be beneficial				x		
Teaching/coaching testing techniques that AREN'T traditional test case and script models			x			
Tediousness of writing mock-driven unit tests				x		
Test awareness in general – usually not taken as seriously as it should be						x
Test data generation and storage	x					
Test redundancy checks						x
Testing frameworks for native C++ are not easy to use and need improvement	x					
Testing is not a career that comes out of post-secondary educational programs						x
Testing team expertise		x	x			
Testing team not given sufficient time		x				
Testing team not involved in direct requirement gathering from client so requirement understanding gap						x
The need for risk and priority based testing						x
Time/effort required to generate tests		x				
Tools are not easily customizable	x					x
Training needs gaps			x			
Verbosity of testing						x

References

Abran, A., Bourque, P., Dupuis, R., Moore, J.W., 2001. Guide to the Software Engineering Body of Knowledge (SWEBOK). IEEE Press.

Ali, S., Yue, T., Briand, L.C., Walawege, S., 2012. A product line modeling and configuration methodology to support model-based testing: an industrial case study. In: MoDELS, pp. 726–742.

Ambler, S., 2009, January. Test driven development (TDD): reality over rhetoric. Dr. Dobb's Journal http://www.ddj.com/architect/212902568?cid=Ambysoft (cited October 2009).

Andersson, C., Runeson, P., 2002. Verification and validation in industry – a qualitative survey on the state of practice. In: Proceedings of the International Symposium on Empirical Software Engineering, pp. 37–47.

Basili, V.R., 1992. Software Modeling and Measurement: The Goal/Question/Metric Paradigm, Technical Report. University of Maryland at College Park.

(Canadian) Consortium for Software Engineering. www.cser.ca (accessed April 2012).

Cater-Steel, A., 2005. Addressing the challenges of replications of surveys in software engineering research. In: International Symposium on Empirical Software Engineering.

Causevic, A., Sundmark, D., Punnekkat, S., 2010. An industrial survey on contemporary aspects of software testing. In: International Conference on Software Testing Verification and Validation, pp. 393–401.

Cordy, J.R., 2003. Comprehending reality – practical barriers to industrial adoption of software maintenance automation. In: Proceedings of the IEEE International Workshop on Program Comprehension, pp. 196–205.

Crispin, L., Gregory, J., 2008. Agile Testing: A Practical Guide for Testers and Agile Teams. Pearson Education.

Cusumano, M.A., Yoffie, D.B., 1999. Software development in internet time. IEEE Computer 32 (10), 60–69.

Engström, E., Runeson, P., 2010. A qualitative survey of regression testing practices. In: International Conference on Product-Focused Software Process Improvement, pp. 3–16.

Garousi, V., 2010a. An open modern software testing laboratory courseware: an experience report. In: Proceedings of the 23rd IEEE Conference on Software Engineering Education and Training, pp. 177–184.

Garousi, V., 2010, June 30. Recent Trends in Software Testing: Opportunities for Industry-Academia Collaborations, Invited Speaker. YouTube Corporation, San Bruno, CA.

Garousi, V., 2010c. Metrics and measurement in software testing. In: Invited Talk and Workshop, Toronto TesTrek Symposium, Organized by the Quality Assurance International (QAI) Global Institute, October 18–20.

Garousi, V., 2011a. Incorporating real-world industrial testing projects in software testing courses: opportunities, challenges, and lessons learned. In: Proceedings of the IEEE Conference on Software Engineering Education and Training (CSEE&T), pp. 396–400.

Garousi, V., 2011, September 20. Better Software Testing Through University/Industry Collaborations, Invited Talk. Calgary Software Quality Discussion Group (SQDG).

Garousi, V., Mathur, A., 2010. Current state of the software testing education in North American academia and some recommendations for the new educators. In: Proceedings of the 23rd IEEE Conference on Software Engineering Education and Training, pp. 89–96.

Garousi, V., Varma, T., 2010. A replicated survey of software testing practices in the Canadian Province of Alberta: what has changed from 2004 to 2009? Journal of Systems and Software 83 (11), 2251–2262.

Garousi, V., Zhi, J., 2012, October. A Survey of Software Testing Practices in Canada (2010). www.softqual.ucalgary.ca/projects/2010/Survey_Software_Testing_Practices_in_Canada (accessed October 2012).

Garvin, D.A., 1988. Managing Quality: The Strategic and Competitive Edge. Free Press.

Gelperin, D., Hetzel, B., 1988. The growth of software testing. Communications of the ACM 31 (6), 687–695.

Geras, A.M., Smith, M.R., Miller, J., 2004. A survey of software testing practices in Alberta. Canadian Journal of Electrical and Computer Engineering 29 (3), 183–191.

Grindal, M., Offutt, J., Mellin, J., 2006. On the testing maturity of software producing organizations. In: Testing: Academia & Industry Conference-Practice And Research Techniques.
Hayes, M., 2002. Quality first. Information Week 889, 38–54.
Iberle, K., Bartlett, S., 2001. Estimating tester to developer ratios (or not). In: Pacific Northwest Software Quality Conference, http://www.stickyminds.com/s.asp?F=S6174_ART_2
IBM, 2012. IBM CASCON (Centre for Advanced Studies Conference). www.ibm.com/ibm/cas/cascon (accessed April 2012).
Jia, Y., Harman, M., 2011. An analysis and survey of the development of mutation testing. IEEE Transactions on Software Engineering 37 (5), 649–678.
Jolly, S.A., Garousi, V., Eskandar, M.M., 2012. Automated unit testing of a SCADA control software: an industrial case study based on action research. In: IEEE International Conference on Software Testing, Verification and Validation (ICST), pp. 400–409.
Kasurinen, J., Runeson, P., Riungu, L., Smolander, K., 2011a. A self-assessment framework for finding improvement objectives with ISO/IEC 29119 test standard. In: European Conference on Software Process Improvement, pp. 25–36.
Kasurinen, J., Taipale, O., Smolander, K., 2011b. How test organizations adopt new testing practices and methods? In: International Conference on Software Testing, Verification and Validation, pp. 553–558.
Koskela, L., 2007. Test Driven: TDD and Acceptance TDD for Java Developers. Manning Publications.
Martin, D., Rooksby, J., Rouncefield, M., Sommerville, I., 2007. 'Good' organisational reasons for 'bad' software testing: an ethnographic study of testing in a small software company. In: International Conference on Software Engineering, pp. 602–611.
Mathur, A.P., 2008. Foundations of Software Testing. Pearson Education.
Ng, S.P., Murnane, T., Reed, K., Grant, D., Chen, T.Y., 2004. A preliminary survey on software testing practices in Australia. In: Proceedings of Australian Software Engineering Conference, pp. 116–125.
Pinheiro, C., Garousi, V., Maurer, F., Sillito, J., 2010. Introducing automated environment configuration testing in an industrial setting. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, Workshop on Software Test Automation, Practice, and Standardization, pp. 186–191.
Quality Assurance Institute, 2002, March. Status of Software Testing. www.geocities.com/mtarrani/StatusOfSoftwareTesting.doc (cited October 2009).
Rubin, H., Yourdon, E., Battaglia, H., 1995. Industry Canada: Worldwide Benchmark Project. Rubin Systems Inc.
Runeson, P., 2006. A survey of unit testing practices. IEEE Software 23 (4), 22–29.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2), 131–164.
Runeson, P., Andersson, C., Host, M., 2003. Test processes in software product evolution – a qualitative survey on the state of practice. Journal of Software Maintenance and Evolution: Research and Practice 15, 41–59.
Scheaffer, R.L., Mendenhall, W., Ott, R.L., Gerow, K., 2011. Elementary Survey Sampling. Cengage Learning.
Schwaber, K., Beedle, M., Martin, R.C., 2001. Agile Software Development with Scrum. Prentice Hall.
Software Engineering Research Group (SERG), 2007. Workshop on Enhancing Software Engineering Practices among Alberta's Industry, Government, and Universities in Alberta (SEAB) , http://www.serg.ucalgary.ca/SEAB/2007 (accessed April 2012).
Statistics Canada, 2007. Software Development and Computer Services Service Bulletin (Data for Year 2007). http://dsp-psd.pwgsc.gc.ca/collection_2009/statcan/63-255-X/63-255-x2009001-eng.pdf (accessed February 2010).
Taipale, O., Smolander, K., Kälviäinen, H., 2005. Finding and ranking research directions for software testing. In: European Conference on Software Process Improvement, pp. 39–48.
G. Tassey, 2002. The Economic Impacts of Inadequate Infrastructure for Software Testing, Planning Report 02-3. Prepared by RTI for the National Institute of Standards and Technology (NIST). http://www.nist.gov/director/prog-ofc/report02-3.pdf
The Standish Group, 2001. Extreme CHAOS. http://www.standishgroup.com/sample_research/showfile.php?File=extreme_chaos.pdf (cited October 2009).
The Standish Group, 2009, October. CHAOS Manifesto. https://secure.standishgroup.com/newsroom/chaos_manifesto.php (cited October 2009).
TMMi Foundation, 2010. TMMi Reference Model v2.0. http://www.tmmifoundation.org/downloads/tmmi/TMMi%20Framework.pdf (accessed February 2010).
Torkar, R., Mankefors, S., 2003. A survey on testing and reuse. In: Proceedings of IEEE International Conference on Software: Science, Technology and Engineering, pp. 164–173.
US Government Accounting Office, 1983. Greater Emphasis on Testing needed to Make Computer Software more Reliable and Less Costly, Report # GAO/IMTEC-64.2.
Whittaker, J., 2008, December. Google vs. Microsoft, and the Dev:Test Ratio Debate. http://blogs.msdn.com/james_whittaker/archive/2008/12/09/google-v-microsoft-and-the-dev-test-ratio-debate.aspx (cited October 2009).
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., 2000. Experimentation in Software Engineering: An Introduction. Kluwer International Series in Software Engineering.
Wojcicki, M.A., Strooper, P., 2006. A state-of-practice questionnaire on verification and validation for concurrent programs. In: Proc. of workshop on Parallel and Distributed Systems: Testing and Debugging, pp. 1–10.

Vahid Garousi (Ph.D., P.Eng.) is a Visiting Professor of Software Engineering in the Middle East Technical University (METU) in Ankara, Turkey. He is also an Associate Professor at the University of Calgary, Canada. He is also a practicing software engineer and coach, and regularly provides consultancy and corporate training services in the North America in the areas of software testing and quality assurance, model-driven development, and software maintenance. During his career, Vahid has been active in initiating several major software testing and software engineering projects in Canada with the Canadian and multi-national software companies. Vahid completed his Ph.D. in Software Engineering in Carleton University, Canada, in 2006. His Ph.D. work was on performance testing of real-time software systems. He has been involved in different software engineering conference committees as an organizing or program committee member, such as ICST, ICSP, CSEE&T, and MoDELS. He is a member of the IEEE and the IEEE Computer Society, and is also a licensed professional engineer (P.Eng.) in the Canadian province of Alberta. He has been selected a Distinguished Visitor (speaker) for the IEEE Computer Society's Distinguished Visitors Program (DVP) for the period of 2012–2014. Among his awards is the Alberta Ingenuity New Faculty Award in June 2007.

Junji Zhi is a Masters student in the Department of Computer Science, University of Calgary. His research interests include: Optimization of Software Documentation and Maintenance Costs, Software testing, Empirical Software Engineering, and Software Release Planning.