A Replicated Survey of Software Testing Practices in the Canadian Province of Alberta: What has Changed from 2004 to 2009?

Vahid Garousi, Tan Varma
Software Quality Engineering Research Group (SoftQual)
www.softqual.ucalgary.ca
Department of Electrical and Computer Engineering, University of Calgary
2500 University Drive NW, Calgary, AB Canada T2N 1N4
{vgarousi, tvarma}@ucalgary.ca

Abstract. Software organizations have typically de-emphasized the importance of software testing. In an earlier study in 2004, our colleagues reported the results of an Alberta-wide regional survey of software testing techniques in practice. Five years after that first study, the authors felt it is time to replicate the survey and analyze what has changed and what not from 2004 to 2009. This study was conducted during the summer of 2009 by surveying software organizations in the Canadian province of Alberta. The survey results reveal important and interesting findings about software testing Practices in Alberta, and point out what has changed from 2004 to 2009 and what not. Note that although our study is conducted in the province of Alberta, we have compared the results to few international similar studies, such as the ones conducted in the US, Turkey, Hong Kong and Australia, The study should thus be of interest to all testing professionals world-wide. Among the findings are the followings: (1) Almost all companies perform unit and system testing with a slight increase since 2004, (2) Automation of unit, integration and systems tests has increased sharply since 2004, (3) More organization are using observations and expert opinion to conduct usability testing, (4) The choices of test-case generation mechanisms have not changed much from 2004, (5) JUnit and IBM Rational tools are the most widely used test tools, (6) Alberta companies still face approximately the same defect-related economic issues as do companies in other jurisdictions, (7) Alberta software firms have improved their test automation capability since 2004, but there is still some room for improvement, and (8) Compared to 2004, more companies are spending more effort on pre-release testing.

TABLE OF CONTENTS

1 INTRODUCTION

An international survey of software engineering practices sponsored by Industry Canada in 2002 [1] 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 the survey, the software produced by the Canadian software industry is rated at the low end of the software quality spectrum in terms, and that it requires considerable effort in defect fixing [2].

Another notable finding from the Industry Canada study was that the largest defect rate per thousand lines of code is in the "telecommunications equipment" industry. Data from two statistical reviews by Industry Canada indicate that this industry has generated the largest revenues in information and communication technology (ICT) manufacturing in the period of 1997-2007. It was also in the top three industries in that sector in terms of employment, had the largest research and development expenditures,

and was in the top three in terms of exports [3, 4]. It is thus evident that the telecommunications equipment industry is an important one in the ICT sector.

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 infrastructure as well as fast turnaround times and short release intervals. In particular, the rising expectations of software consumers will require higher quality standards [2].

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 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, which publishes a report series called CHAOS on the topic.

According to the 2001 CHAOS report [5] 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., cancelled 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 [6]: 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 [7]. On top of this, surprisingly, software testing currently consumes, on average, about 80% of the total cost of software development [8]. 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 [2], our colleagues, Geras et al., described the results of a regional Alberta-wide survey of software testing and software quality assurance techniques in practice. Five years after that first study, we felt it is time to replicate the survey and analyze what has changed and what not. This study was conducted during the summer of 2009 by surveying software organizations in the Province of Alberta.

The remainder of this article is structured as follows. A survey of the related work is presented in Section 2. We describe the structure of the survey in Section 0, and in Section 4 we analyze the survey results. Finally, in Section 5, we draw conclusions, and suggest areas for further research.

2 RELATED WORKS

There are several survey studies on the subject of software testing practices. Our literature search identified the following works [2, 9-14], which are discussed briefly next.

The works in [10, 12] are considered two of the first works on the topic. Gelperin and Hetzel report in [10] the growth and industry penetration of software testing. That survey was filled 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 in [10] 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 not executed) achieved during testing (only 5% viewed this as 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 [10] 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 something. Some observers believe that general industry practice is much worse than the survey profile. This view is supported in [10] by comparing the statistics to an earlier 1983 study by the US Government's accounting office on software testing practices in federal agencies [12]. A brief version of the comparison is shown in Table 1.

	[10] (Respondents: Testing conference attendees)	[12] (Respondents: US federal agencies)
Record of defects found during testing is maintained	73%	30%
Test plan describing objectives/approach is required	61%	10%
Test procedure is documented in the standards manual	45%	44%
A measurement of code coverage achieved is required	5%	13%

Table 1. Comparisons of two similar studies in the US in 1980's.

The study in [2] is conducted by our colleagues, Geras et al., and describes the results of a regional Alberta-wide survey of software testing and quality assurance techniques in practice in 2004. More details about the 2004 survey are discussion in Section 3.1.

A recent survey about Test-Driven Development (TDD) was conducted in 2008 and its results are published in [11]. 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 towards 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 [15], 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 [11] provides evidence to support the claim that although many groups claim to be using acceptance TDD [15], it is still more common for them to capture requirements specifications in the form of documents or models.

The survey in [9] studies the software testing practices in Australia during years 2003 and 2004. 65 industrial respondents completed the survey. The study finds the most evident barrier 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 signify 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 the authors of [9] is that government and public non-commercial organizations, being public-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.

The survey report published in [13] is compiled by the Quality Assurance Institute based on a survey completed at its international conference on software testing in 2002. Similar to the survey published in [10], the results of this survey are biased, since conference attendees are presumably software testing professionals who recognize the value of testing. The involvement ratios of the survey respondents of [13] in different testing activities are shown in Table 2.

The data in Table 2 also include ratios from another survey on verification and validation for concurrent programs [14], conducted by two Australian researchers. There were 35 worldwide respondents in this study [14] who were selected from the IBM developerWorks Multithreaded Java programming forum. It is interesting to observe that most respondents of [13] conduct system testing, while most of the respondents in [14] reported conducting unit testing.

Table 2-Involvement of the survey respondents [13] and [14] in testing activities.

Testing Type	% of Respondents in [13]	% of Respondents in [14]
System testing	86	51
Unit testing	80	80
Acceptance testing	77	Not surveyed
Integration testing	75	65
Stress testing	60	Not surveyed
Pilot testing	46	Not surveyed
Other types of testing	9	Not surveyed

3 Survey Design and Execution

3.1 The 2004 Alberta Survey

The 2009 survey reported in this article builds on and replicates a 2004 Alberta-wide survey conducted by our colleagues, Geras et al. [2]. About 60 industrial respondents completed the 2004 survey. The 2004 survey results indicated that a high number of Albertan firms do not use any formal means for deciding production readiness. The survey authors [2] believed that this practice adds to the risk that these organizations face of releasing defective software. The low number of test levels used, lack of testing training, and lack of formal stopping criteria for testing combine to increase the risk of releasing defects into production versions of the software product. The study [2] suggested that firms need to diversify their testing strategies to utilize more test levels and offer more training to non-developers.

The 2009 survey does not only replicate the 2004 survey but it also tries to answer, in retrospective, how much the above trends have changed in the five year period. All the questions of the 2004 survey have been repeated in the current survey plus four new questions (discussed next). The results of both 2004 and 2009 surveys are shown and analyzed together in this article to reflect the changes that have occurred in the five year period in the Alberta's software testing industry.

3.2 Survey Design

In order to develop a survey that would adequately cover software testing while at the same time permitting us to economize on the number of questions we were asking, we re-used the 20 questions from the 2004 survey [2] and added four new questions to gather more specific information (details next).

In the earlier 2004 study [2], for the baseline design of the survey, our colleagues used a popular software testing knowledge base, i.e., the Software Engineering Body of Knowledge (SWEBOK) [16]. 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. We have again used the same categorization in

our 2009 survey and this has allowed us to employ familiar, agreed-upon terminology in defining the structure and also questions of the survey.

We carefully reanalyzed the question list from the 2004 survey and selected the following questions in each category and also added four new relevant questions. The criteria we used in designing our survey's question set was to make sure that the questions are relevant to the industry and also that they align with the testing knowledge area of the SWEBOK [16]. We also asked two of our industrial partners to review our questions sets before starting the survey. They provided to us useful feedback in revising the wording of some of our questions and asked us to add two of the new questions.

1. Test levels
   - Have you received any formal software-testing related training?
   - Does your organization have a training program that specifically targets any of the following?
   - In your current or most recent software project, did the team conduct the following tests?
   - In your current or most recent project, did the team automate any of the tests?
   - What forms of usability testing are commonly employed in your organization?
2. Test techniques
   - (new in the 2009 survey) Which testing tools and frameworks do you use in your company?
   - In your current or most recent software project, what mechanisms did the team use to generate test cases?
   - Which of the following defect prevention techniques are regularly utilized in your organization?
3. Test-related measures
   - In your current or most recent software project, did the team use any of the following measurements as a guide to planning or designing the tests?
4. Test process management
   - In your current or most recent project, was the testing environment separate from the development or production environments?
   - Please identify the ratio of developers to testers in your current or most recent project.
   - What percentage of the pre-release work effort is spent on testing?
   - Please rank the following defect types by the effort required to fix that type of defect.
   - What criteria does your organization utilize to terminate the testing phase?
   - (new in the 2009 survey) What barriers do you believe prevents your company from adopting testing methodology and testing tools?
   - What barriers do you believe prevents your company from providing software training to testing staff?

Furthermore, to capture the respondents' profiles, we included the following five questions in the survey:

- What best describes your current position?
- On what activities do you currently work?
- Please summarize your company and the type of projects you do in a few keywords.
- (new in the 2009 survey) What is the size of your company (number of employees)?
- (new in the 2009 survey) Which programming languages do you use in your company?

Test 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 product under test. Our survey aimed to determine the extent to which companies are using the testing stages, and to identify the product characteristics that they are currently testing for. Test techniques are mechanisms for identifying test cases and include a wide variety of methods, from ad-hoc to formalized call graphs. Our survey sought to identify the mechanisms that software organizations are using (and the ones they are not using).

Test-related measures being used by organizations were also investigated. These include measurements such as the number of defects in a defect-tracking system, and complex reliability engineering measures. This area 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" [17]. Extreme programming uses "yesterday's weather" in order to assist the team in detailed iteration planning.

Similar to our colleagues' earlier work, we also sought to understand the ways in which organizations manage their testing effort. 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 theory with a number of questions related to managing the test process. All the survey questions (24 in total) as sent to the respondents are available online [18].

3.3 Survey Execution and Respondent Population

According to the latest Software Development and Computer Services service bulletin of Statistics Canada, as of 2007 [19], there were 2,947 software engineers/developers as "paid employees" and 212 software development establishments (firms) in Alberta. The ratios of these numbers to the national statistics are shown in Table 3. The profit margins of its software development firms are higher than the national average. This is perhaps due to the fact that Alberta is an energy-rich province and has higher GDP per capita than the national average. Alberta hosts 10.96% of Canada's population, however 14.06% of Canada's GDP is generated in Alberta (see the detail statistics in Table 3).

Table 3- Population and economy of Canada versus Alberta.

	Canada	Alberta	Alberta's %
Population (2007)	33,115,000	3,632,483	10.96
Software engineers	39,181	2,947	7.5
Software firms	2,156	212	9.8
Profit margin (percentage)	5.6%	13.2%	7.6% higher than national average
Total GDP (2008)	$1,321,360 million	$185,780 million	14.06
GDP per capita (2008)	$46,441	$74,825	61% higher than national average

Our survey invitation strategy was as follows. In addition to the email list of companies used by the earlier study [2] of our colleagues, we sent email invitations to a selected list of Albertan software companies, which we extracted from "yellow pages" and also through Google searches.

The sample included companies with development offices in Alberta, whether those offices were the company's head office or not. 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 from the supporting academic institutions, researchers agreed to publish only summary information from the survey. Ethics approval for the survey was received in June 2009. For conducting this survey and reporting its results, the authors have benefited from the case study research guidelines presented by Wohlin, Höst et al. [20, 21].

They survey was hosted by an online survey hosting service (www.surveymonkey.com) and was available to the invitees for about one month. 53 respondents from Alberta-based software organizations participated in the survey. According to the statistics in Table 3, there were about 2,947 software engineers in Canada (as of 2007), from which one would expect only a ratio (perhaps half or third) would

actually be testers, be interested in testing or a testing-related survey. Thus, the available participant pool in reality might be narrowed down to about 0.3 x 2,947 ~ 1000 testers in Alberta. Considering that 53 respondents out of this pool contributed to this survey, a rough measure of the participation ration equals 5.3% (53/1000).

The number of survey respondents were coincidentally somewhat close to the two similar studies discussed in Section 2 (60 for [2], and 65 for [9]). It is interesting that most studies of this type end up with such small samples. A break-through future work might be to conduct a much broader survey with a wider participant pool (which is not an easy task).

4 Analysis and Summary of Survey Findings

In the following, profiles of the survey respondents are first discussed. Afterwards, the survey results are presented, based on the testing knowledge area structure as proposed by the SWEBOK [16]: (1) test levels, (2) test techniques, (3) test-related measures, and (4) test process management.

4.1 Profiles of the Samples

4.1.1 Industry Sectors

The respondents who participated in the survey were asked to describe their project or product. Their responses were then mapped, if possible, to the top-level industry classification codes of the North American Industry Classification System (NAICS). The results of the industry classification of the responses in both of the 2004 and 2009 surveys are presented in Figure 1. According to this figure, there was an excellent mix of different project and company profiles in the survey data, and this helps our results to be unbiased from certain types of development firms.

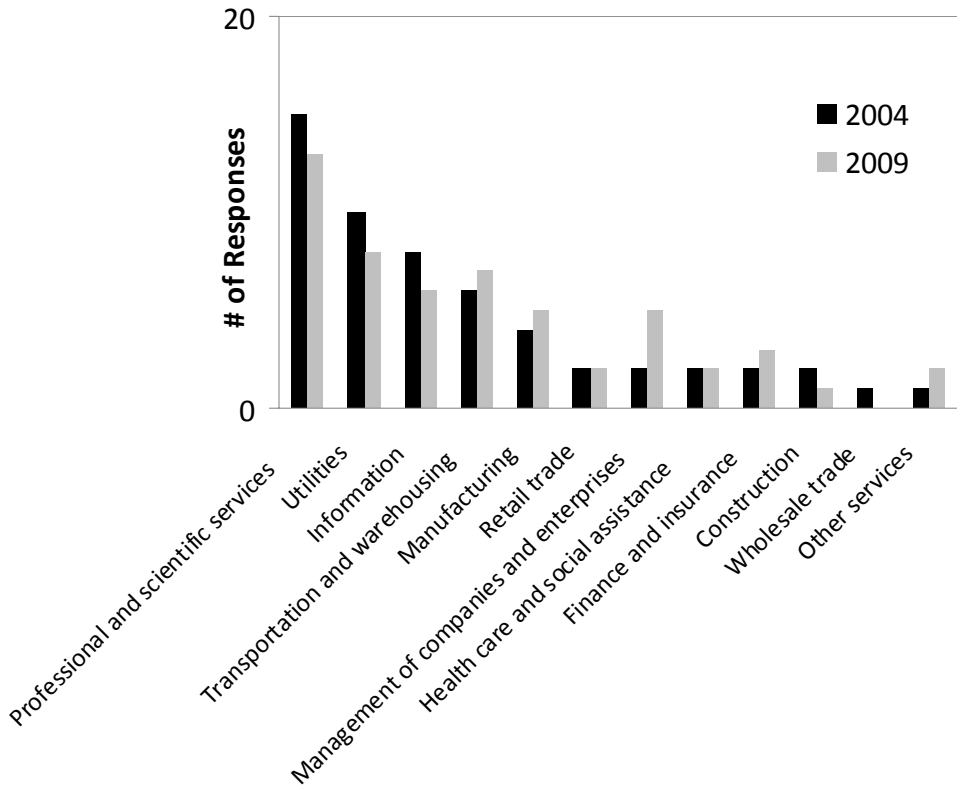

Figure 1- An excellent mix of different project and company profiles in the survey's respondents pool.

4.1.2 Project Types

In term of project types (i.e., external or internal customers), as Figure 2 shows, there was also a fine mix in the survey data. Interestingly, the ratios are somewhat similar to the data from 2004. This seem to provide support and confirmation on the hypothesis that most of Alberta software firm develop software for external customers.

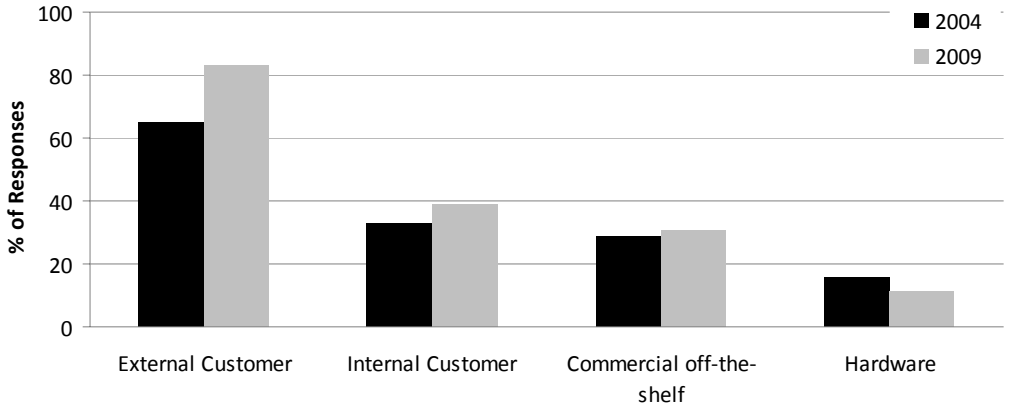

Figure 2-Most of the respondent companies develop software for external customers.

4.1.3 Company Size

The histogram of company sizes in terms of number of employees are shown in Figure 3. Note that this metric was not gathered in the 2004 survey. Most firms had between 10 and 50 employees. It is interesting that a few large corporations with 500+ employees also completed our survey, helping the analysis to cover a wider spectrum of inputs in terms of company sizes.

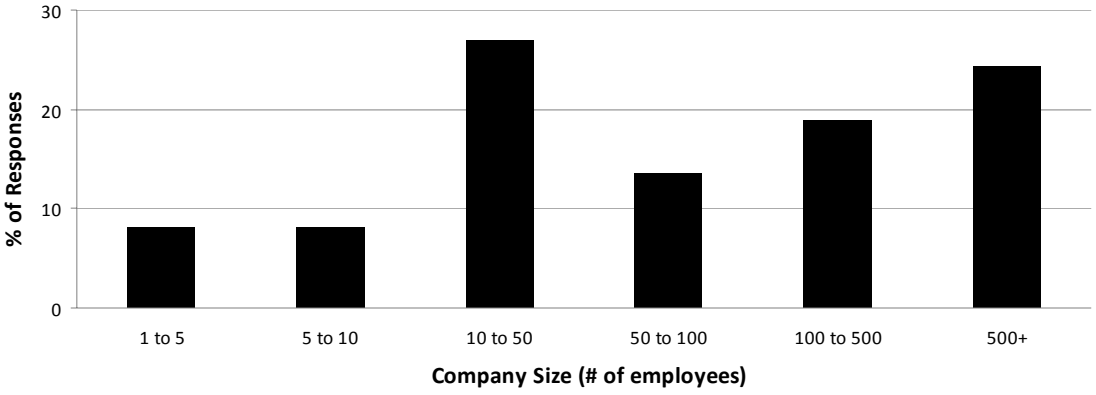

Figure 3-Most firms had between 10 and 50 employees.

4.1.4 Respondents Positions

Respondents' positions in the 2009 study versus the 2004 one are shown in Figure 4. Respondents could provide multiple answers to this question, allowing examples of role overlap to be recorded, e.g., a person could be a team leader and a project manager at the same time. Most of the 2009 respondents were test leads and developers.

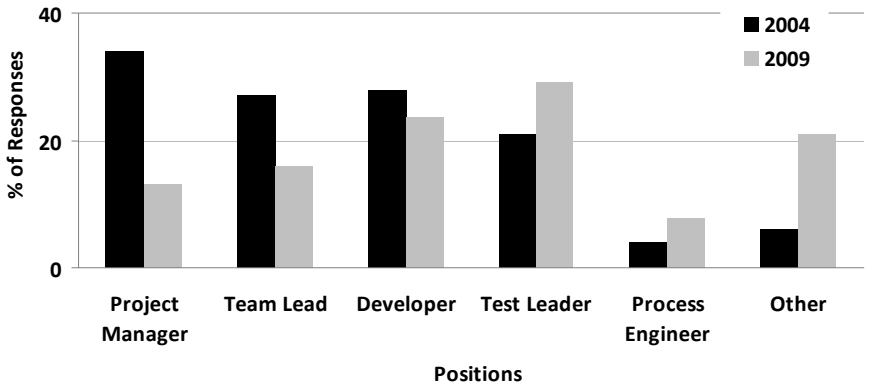

Figure 4-Most of the 2009 respondents were test leads and developers.

4.1.5 Programming Languages used for Development

The histogram of programming languages used by the firms for software development as of 2009 is shown in Figure 5. Note that this metric was not gathered in the 2004 survey and was thus unique to the 2009 survey. Most firms use Java and C++, followed by C#. Informal discussions with our industrial partners (who were also survey respondents) conformed that other programming languages include Visual Basic, PHP, Assembly and other domain-specific languages.

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) have strong automated test framework support (e.g., JUnit) and also industry standard development environments (such as the Eclipse).

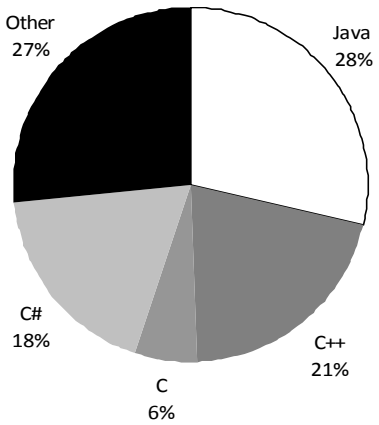

Figure 5- Most firms use Java and C++.

4.2 TEST LEVELS

Test 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 product under test. Our aim was to determine the extent to which companies are using the testing stages, and to identify the product characteristics for which they are currently testing. The percentage of respondents indicating that their organization was using the specified testing stage in their projects is presented in Figure 6.

Almost all companies perform unit (96%) and system testing (97%). As one would expect, unit, system and integration testing have become more widespread since 2004. Other testing levels have different

penetration outcomes in 2004 versus 2009, which might be due to bias from different types of companies completing the survey.

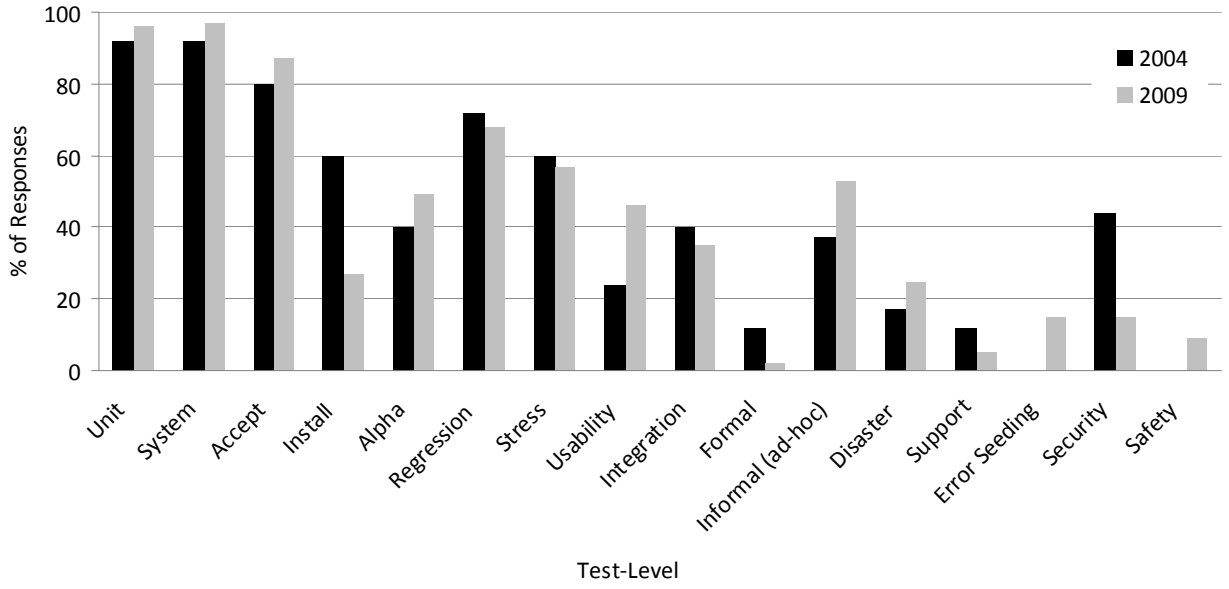

Figure 6-Almost all companies perform unit and system testing.

It is interesting to compare our results to a 2003 survey on the state of the practice for software development worldwide [22]. The survey [22] reports that 91%, 96%, 71%, and 77% of the companies in India, Japan, US, and Europe, respectively, use regression testing in their projects. Our results interestingly show that the usage of regression testing in Alberta (about 70%) is quite similar to the US (71%).

4.2.1 Test-level automation

Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.

We wanted to measure the ratio of test automation to manual testing. Although Figure 6 depicts the scale of conducting different test levels, however it is not clear how much of the tests are automated or manual. Based on the results of a specific survey question on test automation, the ratios of test automation for five main test levels are shown in Figure 7. It is evident that the automation of unit, integration and system tests has increased since 2004. This is a good sign for the Alberta software industry.

By comparing to Figure 7 with Figure 6, one can see that most (about 65%) of the companies who did unit testing as of 2009, conducted automated unit testing, while the rest (about 30%) conducted manual unit testing. Similar comparisons can be made for system, integration and other testing levels.

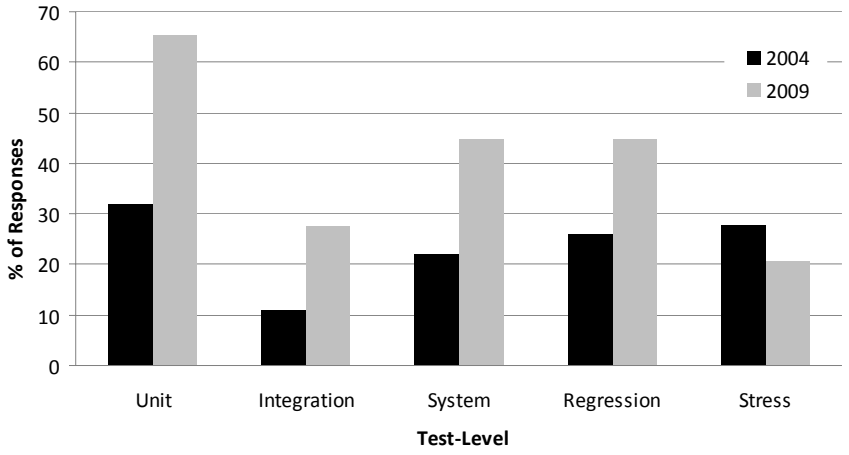

Figure 7-Automation of unit, integration and systems tests has increased since 2004.

To put Alberta's test automation status in context, we would need to compare our data with similar studies. The American Quality Assurance Institute survey of software testing conference attendees yielded results [13] which denote more wider usage of test automation than our survey participants. We also compare our ratios to the study on testing practices for concurrent systems [14]. The comparisons are shown in Figure 8.

Recall from Section 2 that the studies in [13] and [14] had focused participants who were mostly testing professionals. In contrast, we sent our Alberta survey invitations to diverse software development companies in many business sectors without qualifying their level of interest in software testing beforehand. However, the comparison does at least provide some context for the results of the Alberta survey.

It can also be seen from Figure 8 that the Albertan software firms have improved their test automation capability since 2004, but there is still some room for improvement.

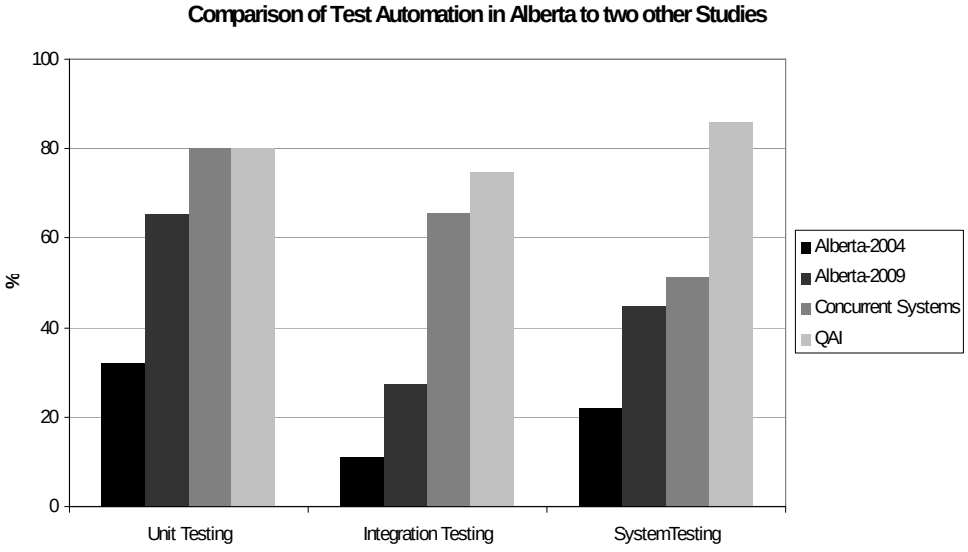

Figure 8-Alberta software firms have improved their test automation capability since 2004, but there is still some room for improvement.

4.2.2 Usability testing

Usability testing is the process of uncovering problems that a user may encounter while using the product. The study in [23] reports that e-commerce businesses lose almost half their potential online business because of usability issues with the e-commerce websites. It is an example of a test level that focuses on the objectives of the test as opposed to the target of the test, in that it focuses on evaluating the ease of using and learning the system [24]. Much of our knowledge of usability comes simply from having customers try out the product. More Alberta organizations are using observations and expert opinion to conduct usability testing (Figure 9). Note that experiments were used in only 34% of the time.

The trend has slightly changed since 2004 and it seems that expert opinions are more preferred lately. This might be due to the increased popularity of the agile practices which usually favor expert knowledge to formal experiments. This result is also consistent with other survey results indicating that organizations tend to prefer informal, low-cost methods over higher-cost, more formal methods even though they rank the higher cost methods as being more important [25].

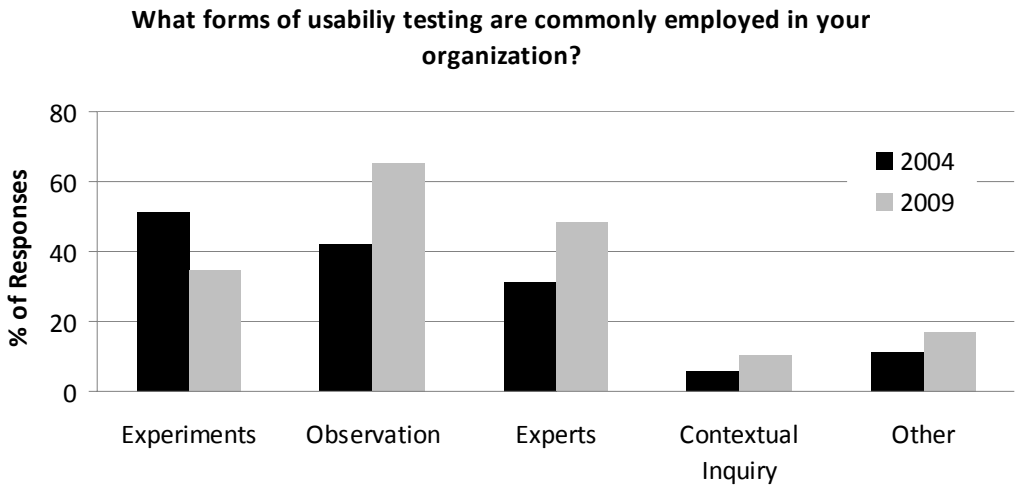

Figure 9-More organization are using observations and expert opinion to conduct usability testing.

4.2.3 Test-level training

According to the 2004 survey [2], the most common type of testing training is at the unit testing level (Figure 10), and correspondingly there was little or no training on testing at the higher levels. For 2004, this meant that there could be a gap in testing skills to overcome should those organizations attempt to adopt test automation at those higher levels, such as system or acceptance testing.

However, the situation seems to have improved significantly in 2009 (Figure 10). 52% of the organizations now provide training on unit testing, 41% on system, 23% on integration and regression testing. One reason for such a big positive jump in better training might be due to widespread use of unit testing framework across the software industry and the real need to have formal testing in the last few years.

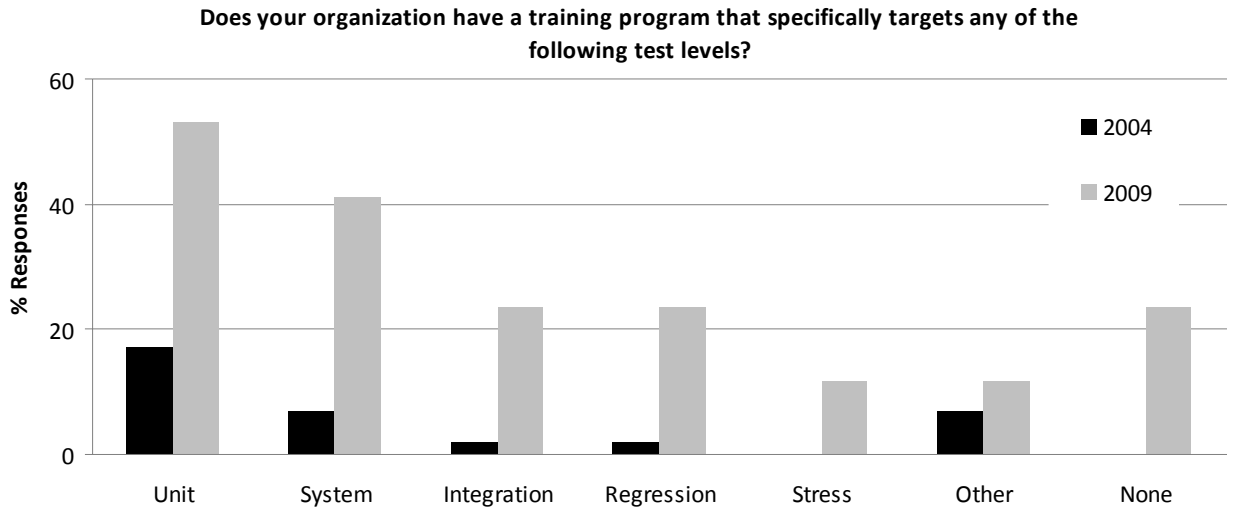

Figure 10- More organizations provide training on unit, system, integration and regression testing.

When the respondents were asked if they themselves had received any formal training on software testing, the answers were interesting (Figure 11). Compared to 2004, fewer managers have received training on testing since they do not usually have to conduct testing themselves. However, more project/test leads and developer have received training on the subject compared to 2004. This is again a positive sign for the Alberta's software industry. But there is still a lot of room for improvement as only less than 20% of the technical staff have received formal training on testing.

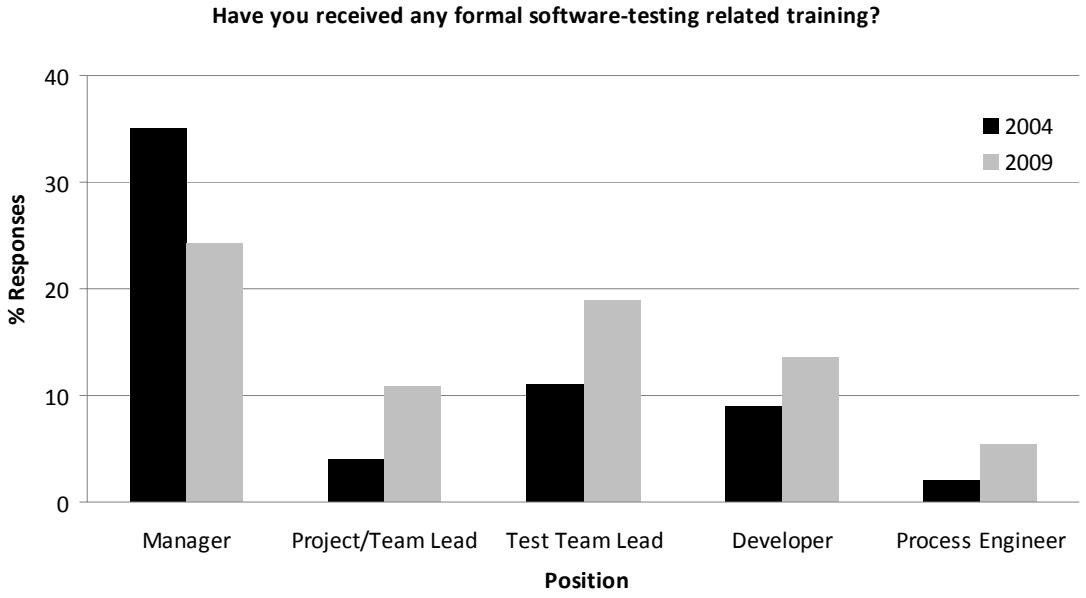

Figure 11-More project, test leads, and developers are receiving formal training on software testing.

To study the root causes of low formal training on testing, our 2009 survey had a question about barriers preventing companies from providing software training to testing staff (results are shown in Figure 12). Note that this question was not present in the 2004 survey. Cost and time are reported as two of the major barriers. It seems that most companies feel that the return on investment (ROI) for software testing training is not very significant and thus they do not provide such training. To mitigate such beliefs, the

Alberta industry needs additional motivational evidence to justify cost expenditure for providing software testing training to their staff.

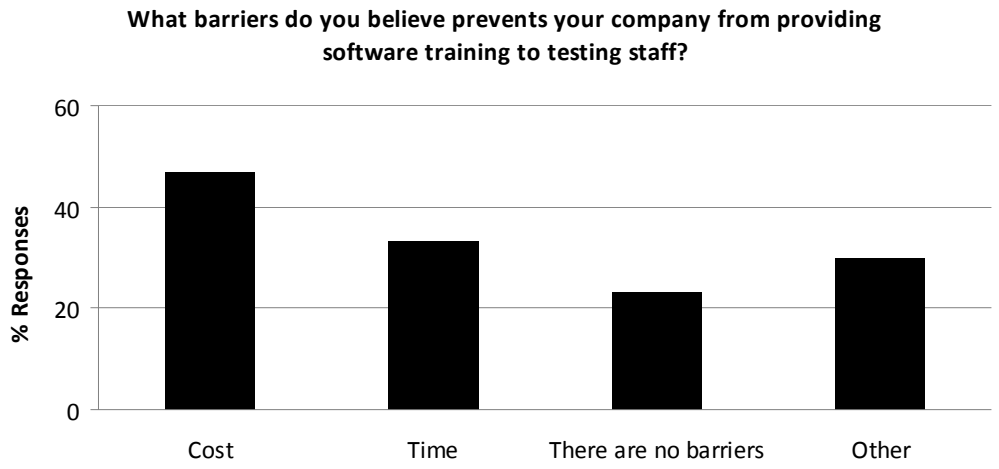

Figure 12- Cost and time are two major barriers for companies in providing software training to their testing staff.

4.3 TEST TECHNIQUES AND TOOLS

4.3.1 Techniques for identifying test cases

Test techniques refer to the methods that software teams use for identifying test cases. In Figure 13, note the extent to which software organizations depended on tester skill and intuition (90%), and the situation has not changed much since 2004. This result contrasts sharply with the relatively low amount of training that organizations offered their personnel (Figure 11).

Most of the firms (86%) supplemented the tester-skill-and-intuition technique with additional testing techniques. This leaves 14% of the firms with a potentially higher risk because of a heavy reliance on their testers, coupled with inadequate training for these people. This may be a "prescription for disaster".

More systematic techniques such as boundary analysis [24], and decision tables seem to have gained slightly more popularity in 2009 compared to 2004.

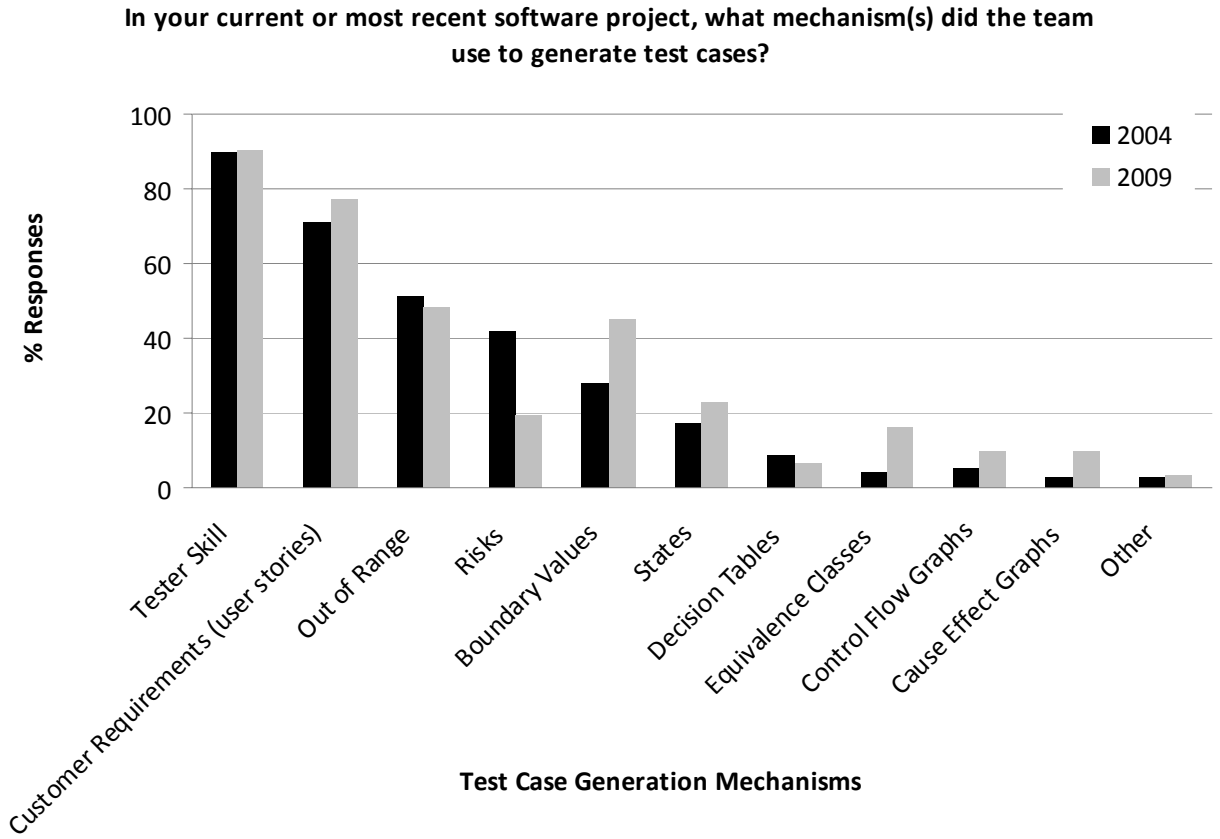

Figure 13-The choices of test-case generation mechanisms have not changed much.

4.3.2 Test techniques for preventing defects

Reuse (26%) and informal inspections (36%) are still the most commonly utilized defect-prevention techniques (Figure 14). However, their popularity has decreased compared to 2004. Three other techniques, namely formal inspections, root-cause analysis and Capability Maturity Model (CMM) have gained popularity during the last five years. This seems to indicate that the Alberta software industry is gaining belief in more systematic method for preventing defects.

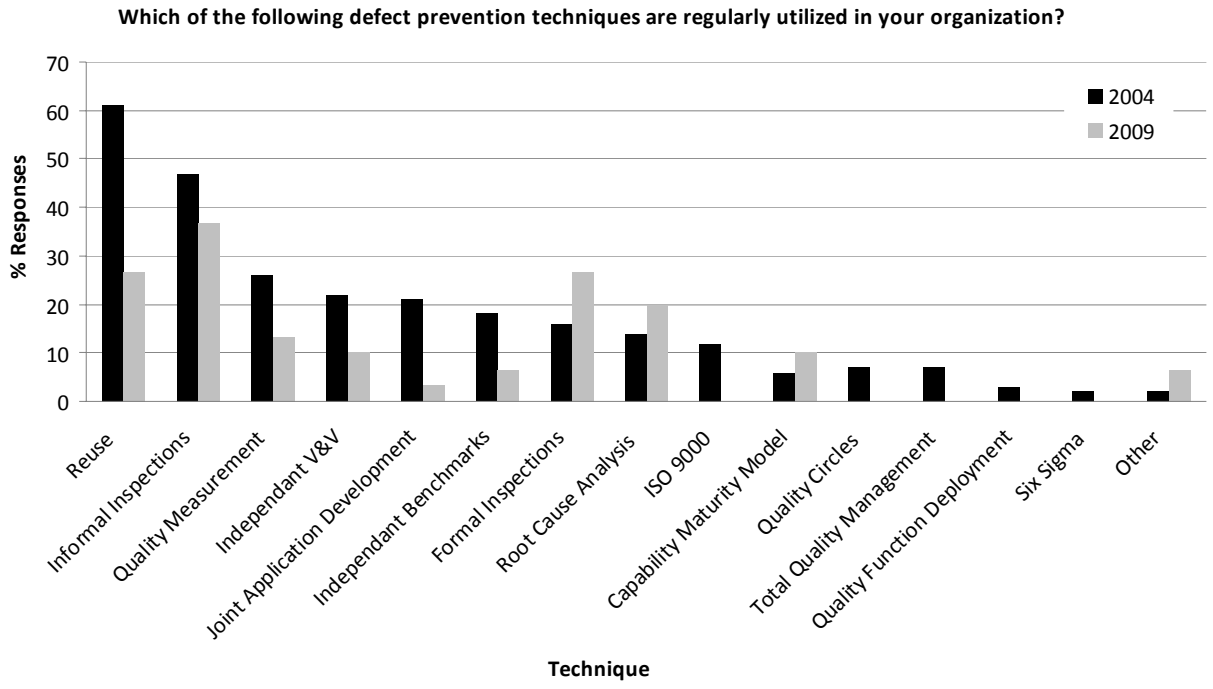

Figure 14-Slightly more project use formal inspections and root-cause analysis for preventing defects.

4.3.3 Test tools and frameworks

One of the new questions in our 2009 survey was the test tools and frameworks used by companies (Figure 15). Note that this question was not present in the 2004 survey. JUnit and IBM Rational testing tools (e.g., Functional tester) are the two most widely used test tools. It can be seen that both open-source free (e.g., the former), and commercial tools (e.g., the latter) are used by the companies. Interestingly, out of 53 respondents, no one reported using the testing products by Parasoft Corporation or Microsoft (e.g., Team Studio).

The trend is consistent with other survey such as the one by Forrester market research firm [26] which ranks the IBM Rational Functional tester tools as the second leading functional testing tool. JUnit is also an extremely popular tool in the global software industry. A consulting firm called Tejas Consulting refers to it as *"probably the most widely used unit test tool on the planet"* [27].

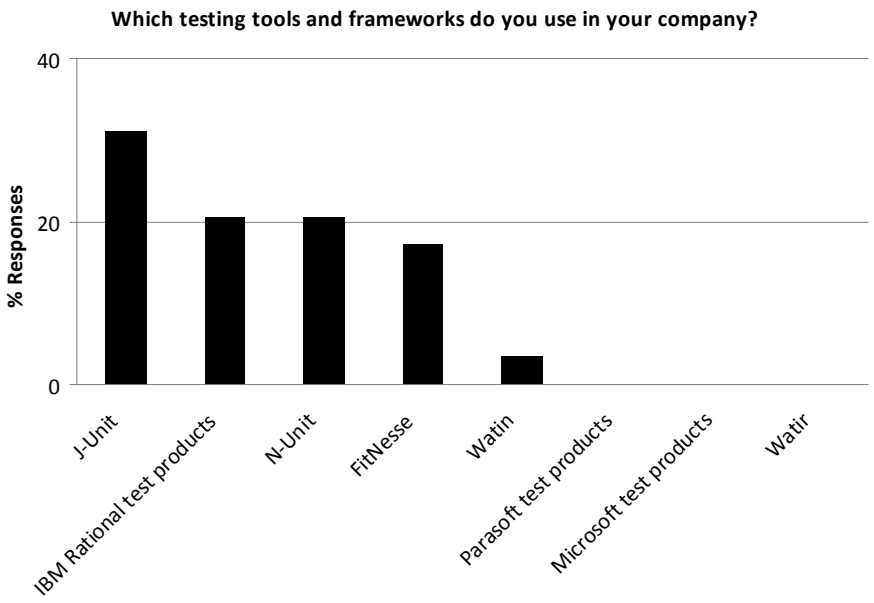

Figure 15-JUnit and IBM Rational tools are the most widely used test tools.

4.4 TEST-RELATED MEASURES

Measurement is instrumental to quality analysis [16]. Therefore it was important to us to determine the extent to which software organizations in Alberta use measurements to plan their testing activities. Almost 72% of the firms reported using use-case points [28] as a planning guide for testing (Figure 16). It seems that the prevalence of use cases in guiding testing effort is noteworthy since it reflects the general industry trend towards using the Unified Modeling Language (UML) and its constituent diagrams. There have been various promising recent works in this area among researchers and practitioners as well (e.g., [28, 29]). This tendency is encouraging, since use cases are requirements artifacts, and using requirements to plan tests is an effective way to confirm that the team has built the right product.

It is interesting that, compared to 2004, more projects are using use-case points as a guide to planning or designing test cases. The use of McCabe complexity and Lines of Code (LOC) has decreased during the passed period.

By comparing our results to those in [30], test measurements in Alberta's software industry seem to be better done that the European industry in 1998. The study in [30] benchmarked the European software management practices and study revealed that while many European organizations emphasize planning of the testing process (e.g., 70% of the UK-based companies), they do not manage the test process with explicit metrics such as test coverage.

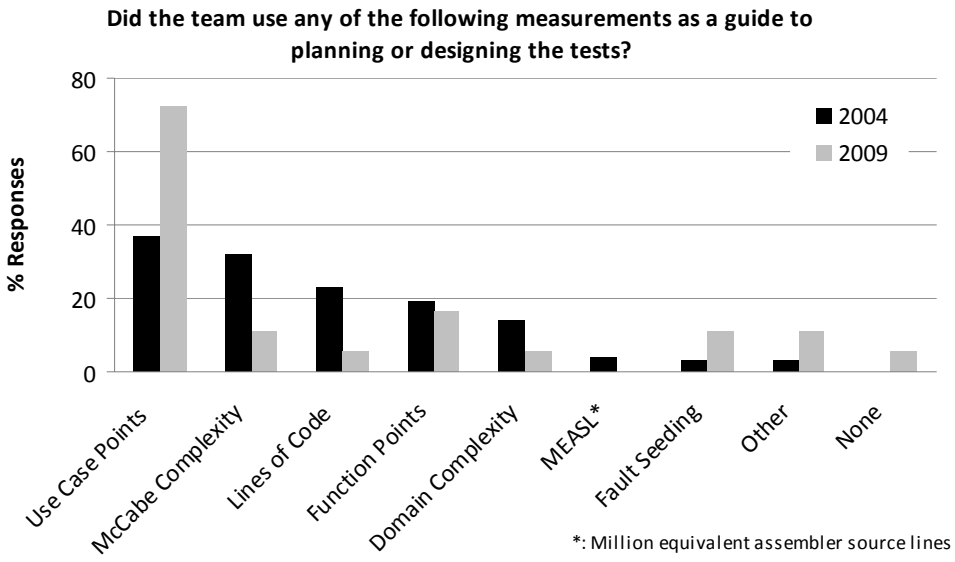

Figure 16-More projects are using use-case points as a guide to planning or designing test cases.

4.5 MANAGING THE TEST PROCESS

4.5.1 Separate test and development environments

The basic question that we were trying to answer with this question was how well funded (well supported) the testing effort is in a typical software organization in Alberta. Having separate hardware and software environments is one of the indicators of a well-funded testing effort. Poorly funded testing efforts would separate only the software environment, while under-funded testing would offer no separate test environment at all.

61% of the responding organizations had separate hardware and software for testing; that is, an environment dedicated to testing software systems (Figure 17). Industry best practices appear to support this separation of testing from development by providing separate facilities and staff for both activities [31].The ratio is slightly different than 2004 which we think is due to the bias in having slightly different types of respondents (i.e., randomness effect). We might be able to say that the testing practices from this perspective has not changed much since 2004.

The results from our study are encouraging, since some two-thirds of respondents reported the testing-development separation, indicating that software development groups take testing seriously.

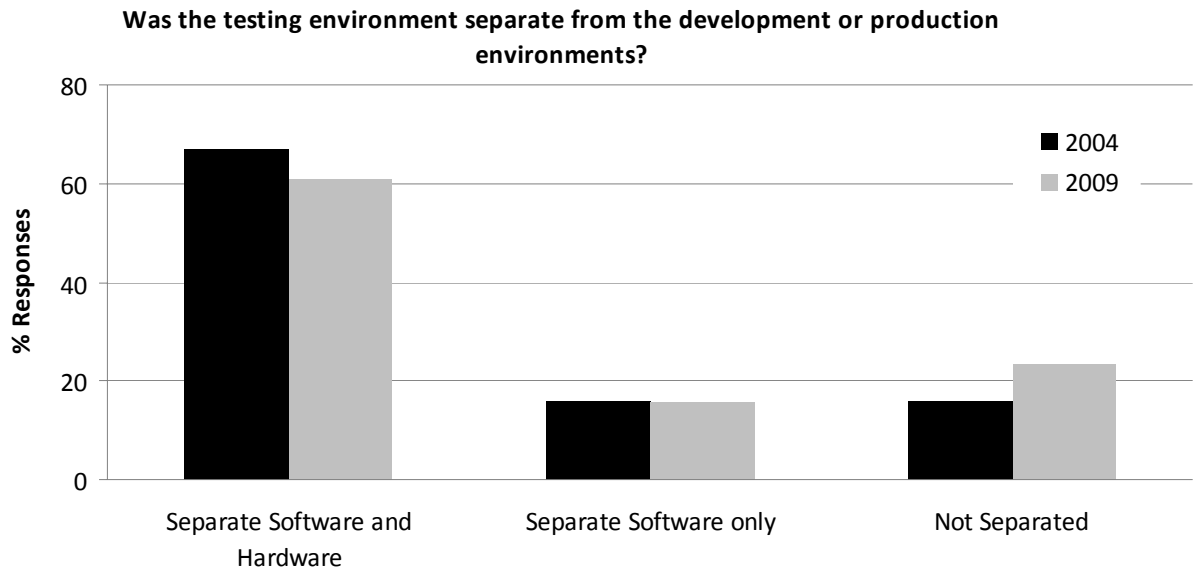

Figure 17-The testing practices with respect to separation of test and development environments have not changed much since 2004.

4.5.2 Relative size of the testing team

In most teams, the number of developers exceeded the number of testers by a wide margin (Figure 18). More than 65% of the teams had at least twice as many developers as testers. Interestingly, this number was almost the same for 2004, but with a different mix, i.e., in 2004, 31% had one tester for two to five developers (1:2 to 1:5), and 35% had less ratios. However, in 2009, 50% had a tester for two to five developers (1:2 to 1:5), and only 15% had less allocation of testers. From comparing this trend, we can say that the Alberta industry is feeling the need for more testers per developer and it is trying to move in the trade-off line towards more testers (see Figure 18).

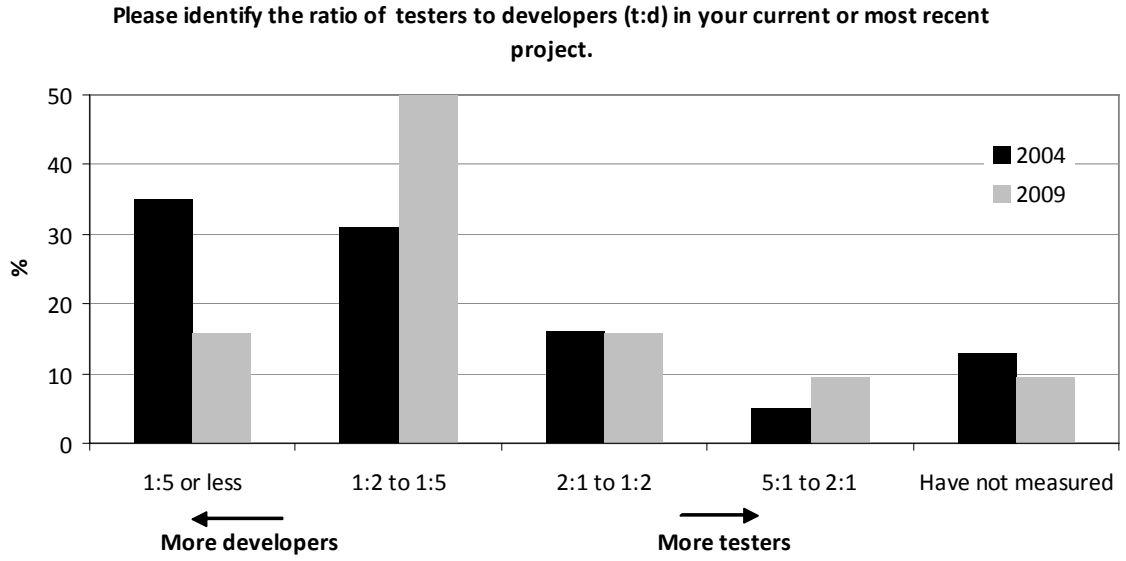

Figure 18-Most of the organizations assign one tester for each 2 to 5 developers.

In 2009, 25% indicated that they had more testers than developers (2:1 to 5:1). This is slightly better than the 21% in 2004.

9% of companies (vs. 13% in 2004) mention that they have not precisely measured how many developers they have per tester, or that perhaps a developer works as a tester too. This is something which certainly needs to be addressed by a company.

The tester to developer ratio (test:dev) is in fact a hot debate in the software industry lately. James Whittaker, a popular test engineer who works for Microsoft, has a recent blog post [32] on the topic where he compares Google vs. Microsoft, in terms for their test:dev ratio. He points out that, at Microsoft, the test:dev ratio varies somewhat from near 1:1 in some groups to 2:1 or 3:1 in others. At Google, just the opposite seems to be the case with a single tester responsible for a larger number of bug-writing developers. He finally concludes that: *"Test managers should be trying to find that sweet spot"* [32]. The authors of [33] also suggest 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 1:2 to 1:5.

There are other studies on the issue of tester to developer ratio. Cusumano and Yoffie [34] 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).

4.5.3 Relative cost to repair defects

The respondents were also asked to "rank the defect types by the (average) effort required to fix that type of defect." The net result was that stress/performance problems were ranked as the most expensive to repair, while requirements problems ranked second (see Figure 19). The results have changed from 2004 and this seems to indicate that performance issues are becoming harder and more expensive to fix. Design and requirements problems are still in top three.

This result is only slightly inconsistent with the results of the Software Productivity Research (SPR) study [35], which only ranked three types of defect types (also appear in Figure 19). These rankings suggest that Alberta companies face approximately the same software defect cost issues as do companies in other jurisdictions.

A 2003 survey of the Turkish software sector [36] also reveals that companies report different costs in repairing defects, however detailed measures are not reported in that survey [36].

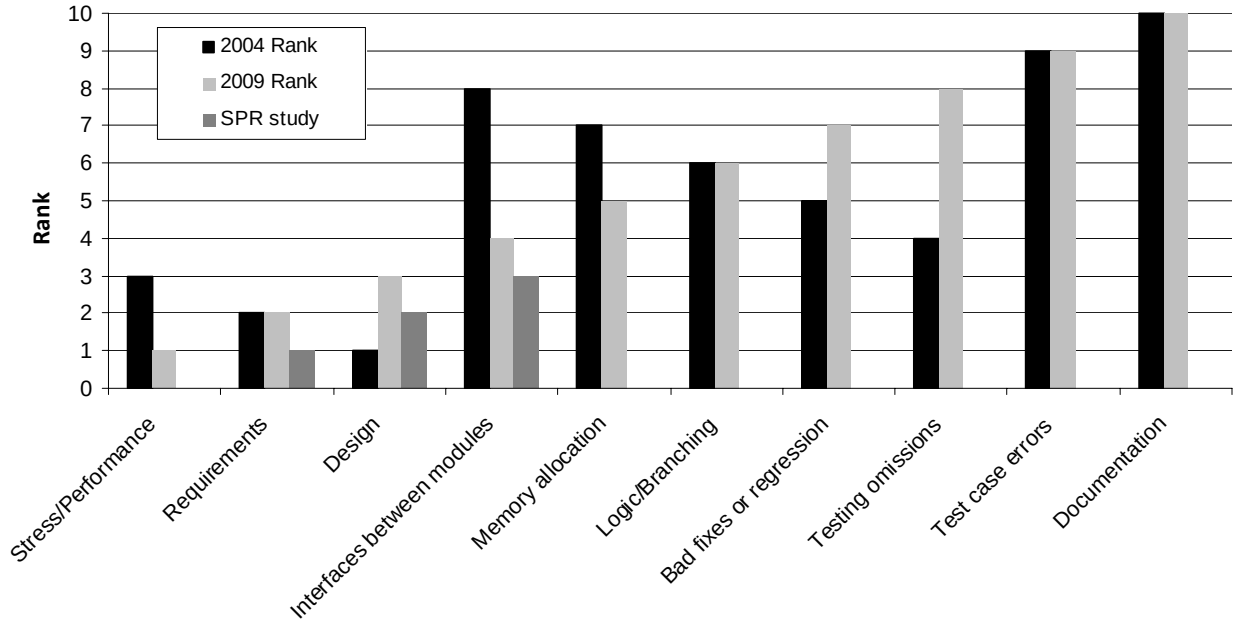

Figure 19- Relative cost to repair defects (ranked by defect types).

4.5.4 Relative effort of pre-release testing

As one might expect, with increasing size and complexity of software systems, the relative effort spent on pre-release testing has increased from 2004 to 2009 (Figure 20). By taking the average values, pre-release testing effort generally accounted for 19.6% of the work in 2004, while this increased to 29.9% in 2009.

A more careful analysis on the rate of increase in size and complexity of software systems versus the reaction of the software industry to spend efforts on testing them is needed.

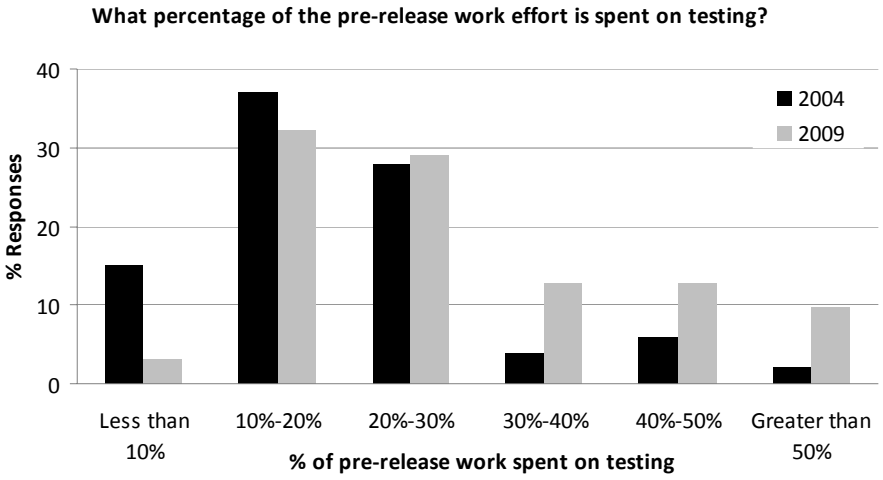

Figure 20-Comparted to 2004, more companies are spending more effort on pre-release testing.

4.5.5 Criteria for terminating the testing phase

The criteria that firms use to decide when to stop testing remain a major problem for most software organizations [24]. 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 comes with a trade-off, making this decision a complex and risky one.

Almost a third of the participating organizations (29%) in our survey indicated that they had no formal means of determining when to stop testing (Figure 21). The majority of firms (70%) used passing acceptance tests as the key indicator for production readiness. This was also the majority in the 2004 survey (58%). While only 29% of the Alberta respondents indicated that they used coverage analysis to stop tests, the KLCI study [37] reported that 44% of the best-practices companies used test coverage, while 29% of the other respondents (the non-best-practice companies) studied in [37] indicated that they also use it. It is good news that, compared to 2004, more Alberta organizations are now using coverage analysis to terminate testing. But still informal criteria are used often which suggests there is still room for improvement on this aspect of testing practices in Alberta.

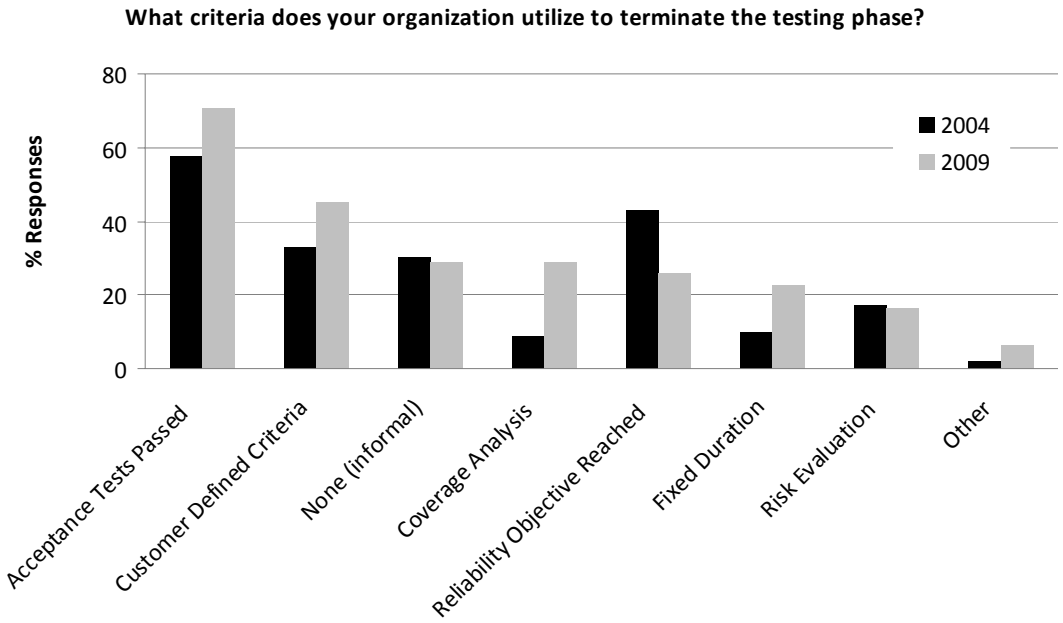

Figure 21-More organizations are using coverage analysis to terminate testing. But still informal criteria are used often.

4.5.6 Barriers for adaptation of testing methodology and tools

The 2009 respondents were also asked about the barriers for adaptation of testing methodology and tools in their companies. Cost, lack of experience, and time were ranked as barriers in order (Figure 22). This question was not present in the 2004 survey. It is interesting to see lack of experience versus cost and time. If a company does not have enough budget and time, there will be no training and it will thus lead to lack of experience in testing. This seems to be a cyclic "chicken-and-egg" problem, which needs to be addressed.

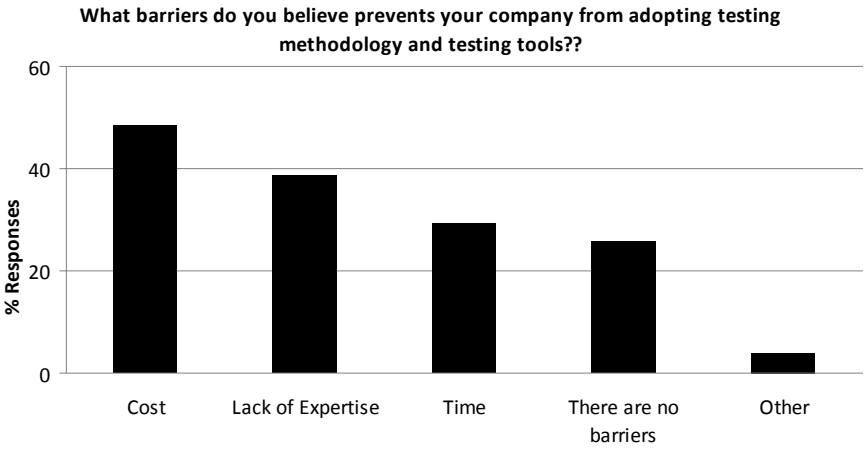

Figure 22-Cost and lack of expertise are two major barriers for adaptation of testing methodology and tools.

5 CONCLUSIONS AND FUTURE WORKS

5.1 CONCLUSIONS

The survey had 53 respondents from firms developing general software systems, software for military applications, software for the oil and gas industry, pharmaceutical applications, travel industry, telecommunications, inspections and facility management, distance education, and also government agencies which develop software. There was an excellent mix of different project and company profiles in our survey data, and this helped our results to be unbiased from certain types of development firms.

Among other results, the survey revealed that inadequate testing training, and lack of formal stopping criteria for testing combine to increase the risk of releasing defects into production versions of software products. 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 US$6.2 billion per year [8]. The relative impact in Canada and Alberta is probably similar, and therefore there is a need for increased quality assurance vigilance. Given the results of the survey, there is a chance that at least a portion of this increased vigilance can take the form of an increase in testing and testing-related activities since, according to our comparisons in general, Alberta companies are doing slightly less testing than companies in other geographic markets. However, fortunately, this has clearly improved from 2004 to 2009. Firms need to diversify their testing strategies to utilize more test levels and offer more training to non developers. Firms that offer testing training to quality assurance, testing, and end-user personnel in addition to their developers will improve their ability to validate quality and correspondingly improve the likelihood of pinpointing and resolving process and product defects.

Few organizations are not offering any form of formalized testing training, yet at the same time they rely heavily on tester knowledge and intuition in order to develop test cases. Further, organizations use passing acceptance tests as the primary criterion for stopping testing. This leads to the question of what the source of testing knowledge in the organization is. While developers get some testing training in postsecondary programs and also receive the bulk of the testing training that firms offer in the workplace, where do the rest of the testing team members get their testing knowledge? Without further study 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.

In the retrospective analysis of what has changed from 2004 to 2009, the survey results revealed important and interesting findings about software testing practices in Alberta. Among the findings were the followings:

- Almost all companies perform unit and system testing with a slight increase since 2004.
- Automation of unit, integration and systems tests has increased sharply since 2004.
- More organizations are using observations and expert opinion to conduct usability testing.
- The choices of test-case generation mechanisms have not changed much from 2004.
- JUnit and IBM Rational tools are the most widely used test tools.
- The testing practices with respect to separation of test and development environments have not changed much since 2004.
- Alberta companies still face approximately the same software engineering economic issues as do companies in other jurisdictions.
- Compared to 2004, more companies are spending more effort on pre-release testing.

- More organizations are using coverage analysis to terminate testing. But still informal criteria are used often.
- Cost and lack of expertise are two major barriers for adaptation of testing methodology and tools.

It is the belief of the authors that general improvement in various testing practice factors from 2004 to 2009 in Alberta should have been partially due to (1) world-wide increased awareness and appreciation of the vitality of systematic testing in the software development lifecycle, (2) possibly due to the results of the 2004 paper, and (3) also a series of several joint industry-academia software testing workshops held by our colleagues at the universities of Calgary and Alberta in 2005 and 2006.

While the replicated survey has shown signs of improvement in various testing practice factors in Alberta, it also reveals that the testing skills and practices of the development teams in Alberta are still maturing.

Some of the survey results (e.g., level of test-level automation and test-related measures) matched quite closely with some other international studies conducted in other countries or regions, (e.g., Turkey, Europe, Australia, and Hong Kong) which revealed similarities in testing practices in Alberta with elsewhere. Some other results revealed different outcomes (e.g., test termination criteria) in Alberta versus other studies which seem to denote that Alberta's software sector as a whole works slightly differently on these test practices, which might be due to its unique characteristics. For example, it is widely believed that the Alberta's software industry mostly serves the regional oil and gas industry and this phenomenon might have implications on the local software engineering practices (e.g., more aggressive schedules and budgets).

The authors also observe that the Alberta industry is paying more attention to the importance of testing lately. This can be observed by increased popularity of several software-quality and testing-related communities in Alberta with frequent meetings and events, e.g., Calgary Software Quality Discussion Group, Calgary Agile Methods User Group, Calgary branch of the American Society for Quality, and Agile Edmonton. 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 method 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.

Furthermore, it is evident Alberta's government agencies are also putting more focus on testing. Informal discussions with the IT professionals in the government of Alberta and also browsing through the latest service procurement notices on the local government websites (e.g., the Alberta Purchasing Connection) seem to suggest that more and more software testing services are being looked for by the Alberta governmental agencies (e.g., departments of energy and education).

5.2 Raising Research Questions

Our survey raises a number of research questions 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 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?
- Why are some of the test levels (e.g., unit and system testing) automated more often than others (e.g., integration) in the industry?
- Since the criteria used by companies for terminating the testing phase are not common, it would be beneficial to empirically evaluate the usefulness of different testing termination criteria in different projects and settings.

5.3 GUIDELINES FOR CONDUCTING SIMILAR SURVEYS

For the researchers or practitioners who are planning to conduct similar surveys, all of our survey questions as sent to the respondents are available online [18].

In inviting the companies to participate, while personal contacts help a lot, our experience has shown that other venues such as yellow pages, Google searches and phone calls are also effective.

Also, our experience has shown that survey questions should be as concise as possible to make sure participants times are used as efficient as possible. It is a good idea to ask several practicing testers to review the questions before making them public to make sure all the questions relate to the real-world practice and use standard terminologies.

5.4 FUTURE WORKS

Further similar studies are needed in other Canadian provinces and also in other 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)* [38].

ACKNOWLEDGEMENTS

Vahid Garousi was supported by the Discovery Grant no. 341511-07 from the Natural Sciences and Engineering Research Council of Canada (NSERC) and also by the Alberta Ingenuity New Faculty Award no. 200600673. Tan Varma was supported by NSERC through the Undergraduate Student Research Awards Program (USRA).

The authors would like to thank Mike Smith, James Miller and Adam Geras for sharing their insights and experience about the 2004 survey and also for providing useful feedbacks on early versions of this article. The authors would also like to thank the anonymous reviewers for their insightful comments in the revision process of this article.

REFERENCES

[1] H. Rubin, E. Yourdon, and H. Battaglia, "Industry Canada: Worldwide Benchmark Project," *Rubin Systems Inc.,* 1995.

[2] A. M. Geras, M. R. Smith, and J. Miller, "A Survey of Software Testing Practices in Alberta," *Canadian Journal of Electrical and Computer Engineering,* vol. 29, no. 3, pp. 183-191, 2004.

[3] Industry Canada, "Information and Communications Technologies (ICT): Statistical Review," *Not available online anymore,* Apr. 2002.

[4] Industry Canada, "Information and Communications Technologies (ICT): Statistical Overview," *http://www.ic.gc.ca/eic/site/ict-tic.nsf/eng/h_it06155.html*, Aug. 2009 [cited: Oct. 2009].

[5] The Standish Group, "Extreme CHAOS," *http://www.standishgroup.com/sample_research/showfile.php?File=extreme_chaos.pdf*, 2001 [cited: Oct. 2009].

[6] The Standish Group, "CHAOS Manifesto," *https://secure.standishgroup.com/newsroom/chaos_manifesto.php,* Oct. 2009 [cited: Oct. 2009].

[7] M. Hayes, "Quality First," *InformationWeek,* vol. 889, pp. 38-54, 2002.

[8] G. Tassey, "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, 2002.

[9] S. P. Ng, T. Murnane, K. Reed, D. Grant, and T. Y. Chen, "A Preliminary Survey on Software Testing Practices in Australia," *Proceedings of Australian Software Engineering Conference*, pp. 116-125, 2004.

[10] D. Gelperin and B. Hetzel, "The Growth of Software Testing," *Communications of the ACM*, vol. 31, no. 6, pp. 687-695, 1988.

[11] S. Ambler, "Test Driven Development (TDD): Reality Over Rhetoric," *Dr. Dobb's Journal*, http://www.ddj.com/architect/212902568?cid=Ambysoft, Jan. 2009 [cited: Oct. 2009].

[12] US Government Accounting Office, "Greater Emphasis on Testing needed to Make Computer Software more Reliable and Less Costly," *report # GAO/IMTEC-64.2*, 1983.

[13] Quality Assurance Institute, "Status of Software Testing," www.geocities.com/mtarrani/StatusOfSoftwareTesting.doc, Mar. 2002 [cited Oct. 2009].

[14] M. A. Wojcicki and P. Strooper, "A State-of-practice Questionnaire on Verification and Validation for Concurrent Programs," *Proc. of workshop on Parallel and Distributed Systems: Testing and Debugging* pp. 1-10, 2006.

[15] L. Koskela, *Test Driven: TDD and Acceptance TDD for Java Developers*: Manning Publications, 2007.

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

[17] K. Schwaber, M. Beedle, and R. C. Martin, *Agile Software Development with Scrum*: Prentice Hall, 2001.

[18] V. Garousi and T. Varma, "A Survey of Software Testing Practices in Alberta," www.softqual.ucalgary.ca/projects/2009/software_testing_practices_in_AB, May 2009 [cited: Oct. 2009].

[19] Statistics Canada, "Software Development and Computer Services Service Bulletin," http://dsp-psd.pwgsc.gc.ca/collection_2009/statcan/63-255-X/63-255-x2009001-eng.pdf, 2007, Last accessed: Feb. 2010.

[20] P. Runeson and M. Höst, "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164, 2009.

[21] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell, *Experimentation in Software Engineering: An Introduction*: Kluwer International Series In Software Engineering, 2000.

[22] M. Cusumano, C. F. Kemerer, and B. Crandal, "Software Development Worldwide: The State of the Practice," *IEEE Software*, vol. 20, no. 6, pp. 2-8, 2003.

[23] J. Nielsen, "Did Poor Usability Kill E-commerce?," *Jakob Nielsen's Alertbox*, http://www.useit.com/alertbox/20010819.html, Aug. 2001 [cited Oct. 2009].

[24] A. P. Mathur, *Foundations of Software Testing*: Pearson Education, 2008.

[25] K. Vredenburg, J.-Y. Mao, P. W. Smith, and T. Carey, "A Survey of User-centered Design Practice," *SIGCHI Conf. Human Factors Comput. Syst*, pp. 471-478, 2002.

[26] C. Schwaber and M. Gilpin, "Evaluating Automated Functional Testing Tools," *Forrester Research*, 2005.

[27] T. Consulting, "Open Testware Reviews - JUnit," http://tejasconsulting.com/open-testware/feature/junit-3.8.1.html, June 2003 [cited: Oct. 2009].

[28] S. Nageswaran, "Test Effort Estimation Using Use Case Points," *Quality Week*, 2001.

[29] C. Nebut, F. Fleurey, Y. L. Traon, and J.-M. Jezequel, "Automatic Test Generation: A Use Case Driven Approach," *IEEE Transactions on Software Engineering*, vol. 32, no. 3, pp. 140-155, 2006.

[30] S. Dutta, L. N. van Wassenhove, and S. Kulandaiswamy, S, "Benchmarking European Software Management Practices," *Comm. ACM*, vol. 41, no. 6, pp. 77-86, 1998.

[31] E. Dustin, *Effective Software Testing: 50 Specific Ways to Improve Your Testing*: Addison-Wesley Professional, 2002.

[32] J. Whittaker, "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*, Dec. 2008 [cited: Oct. 2009].

[33] K. Iberle and S. Bartlett, "Estimating Tester to Developer Ratios (or Not)," *Pacific Northwest Software Quality Conference, http://www.stickyminds.com/s.asp?F=S6174_ART_2*, 2001.

[34] M. A. Cusumano and D. B. Yoffie, "Software Development in Internet Time," *IEEE Computer*, vol. 32, no. 10, pp. 60-69, 1999.

[35] C. Jones, "Software Quality in 1999: What works and what doesn't," *Software Productivity Research, Available upon request from capers@spr.com*, 1999.

[36] T. Aytaç, S. Ikiz, and M. Aykol, "A SPICE-Oriented, SWEBOK-Based Software Process Assessment on a National Scale: Turkish Software Sector Survey - 2001," *3rd International SPICE Conference*, 2003.

[37] P. Kulik and C. Weber, "Software Metrics Best Practices (2001)," *KLCI Research Group, http://java.ittoolbox.com/documents/document.asp?i=1750 (link not active anymore)*, Mar. 2002 [cited Oct. 2009].

[38] TMMi Foundation, "TMMi Reference Model v2.0," *http://www.tmmifoundation.org/downloads/tmmi/TMMi%20Framework.pdf*, Last accessed: Feb. 2010.