# Software Testing Practices in New Zealand

**Paul Won-Bin Sung**
**Department of Information Systems and Operation Management**
**University of Auckland**
Email: wonbin81@gmail.com

**John Paynter**
**Department of Information Systems and Operation Management**
**University of Auckland**
j.paynter@auckland.ac.nz

## Abstract

The importance of software testing has often been overlooked. Testing is now beginning to gain the recognition it deserves within the software development process. However, there is still a lack of commonality within software testing with no common standard or guidelines to which testers can refer.

This study uses a framework developed by Sung (2005) that covers all of the major areas of software testing by integrating the current existing standards. Software development organisations in New Zealand were surveyed. In the survey the general status of software testing in New Zealand is examined. We consider how software development organisational practices match the framework. The results show that organisations in New Zealand are not covering the major aspects of software testing as well as they should. This appears due to the fact that New Zealand organisations are small and do not have the resources to fully implement test procedures.

The findings from this research are useful as they give an indication on the status of testing in New Zealand and what needs to be improved.

*Keywords*: Software Testing, Frameworks, IEEE.

## 1 Introduction

Software quality is a critical factor for software development organisations. Software testing plays an important role in this. It is common to find the testing phase taking up more than 40% of the software development process (Chakraborty and Arthanari 1994).

Software testing is defined as "the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item" (Software Engineering Technical Committee 1983, p.10).

The amount of time spent on testing is increasing due to more diverse and complex nature of software being developed. GUI (Graphical User Interface), hardware, software itself and networking components must all be tested together to ensure that the software runs smoothly (Fournier 1998).

Testing should not happen at the end of the software lifecycle, but should occur in parallel with the software development process (Schach 2002, p.161). That is, you cannot test in quality at the end of a project.

Software testing involves much time and effort even with the use of automated software testing tools. Software testing is a difficult task. It is hard to determine how the software will malfunction when invoked by end users in the real world. It is used to verify and validate that the functional requirements are met (Lewis 2000, p.7). Software testing ensures that the software design, code, and documentation all meet the requirements set by the organisation, or by the external standards set by software testing governing bodies. Thus software artefacts must be tested at the end of each phase of the lifecycle. This verification process ensures that the software is being developed in the right manner, while validation ensures that the right software is being developed (Pressman 2005, p.388).

Conventional software applications are tested from two main perspectives. The first approach is testing the inner logic of the software to make sure the program is executed properly. This is often referred to as "white box" testing. The second approach is testing the requirements of the software and this is commonly referred to as "black box" testing. The functionality is tested without understanding the inner logic of the software (Pressman 2005, p.424). Software development organisations adopt different strategies, but the major objective of testing is to "design tests that systematically uncover different classes of errors and to do so with a minimum amount of time and effort" (Pressman 2005, p.146).

We start by considering the background of the software industry in New Zealand and of software testing in particular. Sung (2005) devised a framework for evaluating software testing, based on a literature review. Use of a common framework will provide unity within the software testing discipline and provide more structure and better comparison among different organisations. In this research software development organisations are surveyed to see how they perform against this framework. The findings of the survey are compared against the framework and the results discussed against those from other studies, specifically its Trans-Tasman survey counterpart (Ng, Murnane *et al*. 2004).

## 2 Background

The software industry in New Zealand is growing (Software New Zealand 2005). Thus, the research objective is to find out the status of software testing in New Zealand and examine how it compares with that overseas.

## 2.1 Standards

Standards affect our lives everyday. Standards guide how work should be done and determine quality of the output produced. Reid (2000) postulates that standards ensure that a certain level of quality is achieved in a given product or service. Standards benefit the consumers from high quality product. They also benefit the producers as they insulate them from legal liability and bad publicity (Reid, 2000). Unfortunately, no common standard governs all aspects software testing. There are standards for different aspects of software testing such as BS (British Standard), IEEE (The Institute of Electrical and Electronics Engineers), ISO (The International Organisation for Standardisation), IEC (The International Electrotechnical Commission), NIST (The National Institute of Standards and Technology), CMM (Capability Maturity Model) and TMM (Testing Maturity Model). However, there is no one single software testing standard that is universal across the industry.

Sung (2005) focused on four standards that influence software testing, BCS (British Computer Society) standards, ISO IEC 90003, IEEE and CMM standards. BCS, IEEE, CMM standards are concerned more with process context of software testing and ISO standards relate more to quality context of software testing. Although BCS scheme is fairly comprehensive, there is still a lack of commonality within the software testing (Reid 2000), as there is no one major standard that governs software testing. The proposed framework (Table 1) attempted to address this need.

The software testing standards not only cover the actual testing process but some apply to the broader discipline of software development. Testing standards should incorporate relevant standards from the bigger area of software engineering to improve its process.

There is little commonality or unification among the software testing standards. Ng *et al.* (2004) stated that many software organisations in Australia did not use the published standards, possibly indicating lack of practicability or applicability across all software organisations. Standards need to be accepted by communities of associated professionals, otherwise they become useless. It is generally regarded that "the best standards in the world are of little value if they are not adopted and used by the professional community for which they are aimed" (Buckley 1984).

## 2.2 Methods and Techniques

Testing methods and techniques can be viewed from three broad perspectives; functional versus structural, dynamic versus static and manual versus automated. Although it can be difficult to carry out all of the possible techniques, organisations must select methods and techniques that best suits them, the product and their client. Testing without a method or technique is referred to as "*ad hoc*" testing. Although "*ad hoc*" testing can be useful,

software organisations need formal testing methodologies and techniques to be effective and have a higher chance of success (Lewis 2000, p.19).

**Table 1: A Framework Software Testing Standard (Sung, 2005)**

| |
|---|
| **Testing Methods and Techniques** |
| - Functional and Structural Testing |
| - Dynamic and Static Testing |
| - Manual and Automated Testing |
| **Testing Tools** |
| - Different Types |
| - Criteria for Selection |
| **Testing Standards** |
| - What to Test |
| - Level of Testing |
| - How to Test |
| **Test Management** |
| - Planning and Design |
| - Configuration Management |
| - Testing Training and Experience |
| - Documentation |
| - Review |
| **Testing Metrics** |
| - Performance Measures |
| - Quality of Software |
| - Criteria to Stop Testing |

## 2.3 Testing Tools

Tool use can make the software testing process much easier and more effective. There are tools available for different phases of the testing process. Tools can assist various stages of the software testing process. Kit (1998, p.52) has identified five different categories of software testing tools; tools for planning, tools for test design and development, tools for test execution and evaluation tools, tools for reviews and inspections, and software testing support tools. Tools for planning do not involve specialised tools, but usually involve the testing team creating a template test plan in a word processing document (Kit 1998, p.146). Various tools in the market assist with scheduling and coordination of staff. Tools for test design and development also do not involve specialised tools, but word processing documents that contain test case design, procedures and data sources. However there are specialised tools to assist with data generation and the requirements-based test design (Kit 1998, p.147). Some tools mainly support execution and evaluation. These tools involve capture/playback, coverage analysis and script management (Kit 1998, pp.147-151). Software testing support tools cover the entire testing process. This includes problem management, change control and configuration management. Finally, tools for reviews and inspection involve complexity analysis, code comprehension and syntax analysis tools. (Kit 1998, p.145)

Testing is a crucial part of software development and should also be aided by tools in order to make it more effective. The tools must be reliable and operate consistently with different loads (Hendrick 1999). An important question in any software development organisation is "Is the product ready for release?"

In order to answer this question software defects must be identified and checked against three dimensions; (Hendrick 1999):

Reliability - Software must function correctly without crashing or causing major failure.

Functionality - Software must meet business requirements and behave in accordance with customer needs.

Performance - Software must perform in a timely manner with appropriate load.

In order to meet these needs, organisations must have integrated testing tools to support and coordinate the process. It is important that the testing tools meet the requirements by software development organisations and that their tools cover the key areas in software testing. Testing tools can simplify the testing process and provide measurements on the status of software testing. Use of software testing tools helps software development organisations to release their product by having more confidence on the quality of their product (Hendrick 1999).

## 3 Methods

Based on the literature review, several hypotheses are proposed (Table 2). The methodology used to test these hypotheses is outlined in this section. The major approach undertaken is a survey. It outlines the objectives, method, survey tool, survey description, sample selection, and pilot testing. As well as the survey, content analysis of the testing tool is carried out and compared against the proposed framework in the following sections.

There are two objectives for this survey. First is to find out about the general status of software testing in New Zealand and existence of significant relationship between variables. Second is to see how well the items in the framework proposed by Sung (2005) are being addressed by current software development organisations in New Zealand. This survey research is considered descriptive. The purpose of descriptive survey research is to investigate situations, events and other phenomena occurring within the population (Pinsonneault and Kraemer, 1993). Although analysis of variables and testing of hypotheses are carried out in this study, its aim is not to establish causal relationships, but to test the existence of relationships between variables. Elements of analytical research are present, as results are compared against the framework, but the survey research has more elements of descriptive research.

The survey is aimed at software development organisations in New Zealand, especially the software testing team. The unit of analysis is organisations, and no sampling procedure is carried out as the number of such organisations in New Zealand is limited.

### 3.1 Survey methodology

A Web-based survey was chosen for this study due to its convenience and greater efficiency (Lee 2002). An email was sent out with a link to participate in the survey. With Internet access common in people's lives and with all of the software organisations in New Zealand possessing an email address, an online survey was favoured over its counterparts. Although there are some drawbacks (Frazer and Lawley, 2000) online surveys are much cheaper and more convenient, both for the participants and the researchers. We examined the following categories; software testing training and experience, software testing methodologies, techniques and procedures, software testing tools, software testing metrics, software testing standards, and other general practices.

## 4 Survey results

### 4.1 Respondent profile

429 organisations were emailed the survey link and 61 usable responses were received.

Many New Zealand organisations were small, 45% had from 1-9 employees although some were large (>10000). Most were in the private sector (75%) and were for technology companies (51%). Overall a large range (sector, size, public/private and other dimensions) of organisations responded to the survey.

Hypotheses derived from the original research questions , as outlined in the BCom Honours Research Essay on which this paper is based (Sung 2005), were analysed (Table 2) using SPSS and are covered in the following sections. The testing methods used are shown in Figure 1. In general, adoption of training, tools, metrics, and standards was not high. Manual testing was still most popular among New Zealand organisations.

Note that in reporting the following results some statistics may sum to more than 100% (or exceed the number of respondents), as multiple responses were allowed.

**Figure 1: Testing methodologies**

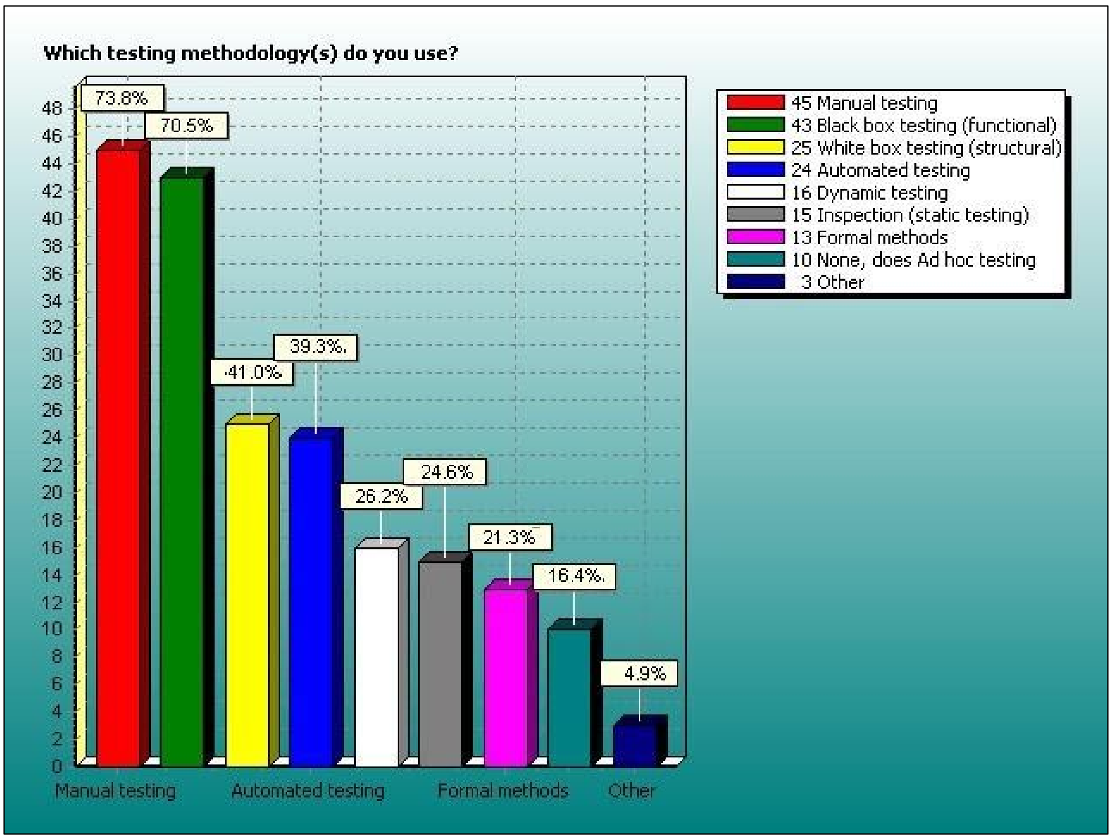

**Table 2: Research Hypotheses.**

| **Hypothesis** | **Statistical test** | **Result** |
|---|---|---|
| **$H1_0$:** Perceived software quality is not related to size of the testing team. | Correlation, regression | Significant at 5% level. Perceived software quality is negatively related to size of testing team. |
| **$H2_0$:** Perceived software quality is not related to amount of testing (proportion in total development). | Correlation, regression | Weak relationship. Significant at 8% level. Perceived software quality is positively related to amount of testing. |
| **$H3_0$:** Perceived software quality is not related to amount of training received by testing staff. | Correlation, regression | Not significant. |
| **$H4_0$:** Perceived software quality is not related to adoption of different methodology(s). | None, cluster report | Unable to comment statistically. No significant differences observed. |
| **$H5_0$:** Perceived software quality is not related to volume of test plans. | Correlation, regression | Not significant. |
| **$H6_0$:** Perceived software quality is not related to volume of documentation. | None, cluster report | Unable to comment statistically. No significant differences observed. |
| **$H7_0$:** Perceived software quality is not related to adoption of different support tool(s). | None, cluster report | Unable to comment statistically. Adoption rate of all three support tools appeared higher when perceived software quality was high. |
| **$H8_0$:** Perceived software quality is not related to adoption of different metric(s). | None, cluster report | Unable to comment statistically. Adoption rate of different metrics appeared higher when perceived software quality was high. |
| **$H9_0$:** Perceived software quality is not related to adoption of different standard(s). | None, cluster report | Unable to comment statistically. No significant differences observed. |

## 4.2 Testing Methods and Techniques

Testing methods and techniques in New Zealand appear to be somewhat restricted largely due to lack of resources and experienced personnel. Many of the organisations adopt manual testing. Although manual testing is a necessary part of software testing, there has been a trend towards automated testing (Schwartz 2003). Black box testing (also known as functional testing) was commonly carried out (i.e. by 85% of the organisations) to make sure that the software worked correctly according to its requirements. Methods such as inspections (25%) and other static techniques were less popular as these methodologies usually take greater time and experience from the testing staff. Due to lack of time and high cost, many (61%) of the small software organisations (defined here as having 0-9 employees) did not seem to fully utilise software testing methodologies. Ten organisations (16%) claimed that they did not have a particular methodology, but did "*ad hoc*" testing instead. Although "*ad hoc*" testing can uncover some major faults, a more structured method is required to detect faults in a systematic manner.

### 4.2.1 Functional and Structural Testing

It was clear from the survey that more organisations did functional testing than structural testing (i.e. 71% performed black box testing rather than the inner logic, white box testing, 41%). The software had to work correctly, so functional testing was considered an important aspect. The functional testing methodology includes functional testing, regression testing, validation testing, system testing and user acceptance testing and these were all common types of techniques carried out by organisations. White box testing (also known as structural testing) was also popular making sure the inner logic of the software behaved in a correct manner. Many developers (i.e. coders) act as testers in software organisations in New Zealand (probably due to the small number of employees), so structural testing, such as unit testing was quite common. Integration, security and recovery testing were also noted by the respondents.

### 4.2.2 Dynamic and Static Testing

Dynamic testing involves execution of the software. This includes black box testing, white box testing and "*ad hoc*" testing (Lewis 2000, pp.22-23). However, not all participants who said they did black box testing also ticked dynamic testing. This may indicate misunderstanding of the terms in the questionnaire or inconsistency within the software testing discipline. This is also evident in the literature where the terms were not being used in a consistent manner. However, it is clear from the survey that dynamic testing was more common than inspections or other static testing methods. Inspections, walkthrough and syntax checking can be quite time consuming and this could be a possible reason for many organisations overlooking this process.

### 4.2.3 Manual and Automated Testing

Although automated testing is becoming more popular, manual testing is still far more commonly practiced in New Zealand (manual, 74%, automated, 39%). Automated testing tools can be quite expensive and for small organisations in New Zealand, it may be more efficient to carry out manual testing. Even if an organisation does automated testing, manual testing should still be done in conjunction, as it can detect major faults within the software. However for repetitive tasks such as regression testing, it will be far more efficient to perform automated testing.

### 4.2.4 Barriers to adopting a testing methodology

Major barriers to adopting a testing methodology appeared to be cost and time. As many (62%) of the software organisations have less than five testing employees, time and money can be scarce resources. Also lack of expertise was the third highest barrier to adopting a methodology, indicating that there is lack of software testing specialists However, the small nature of organisations may mean that people in New Zealand software organisations need to be multi-skilled.

With so much manual testing done across software organisations in New Zealand, testing personnel should be properly trained, the success of manual testing depends on the skill of the testers. However, the results from the survey showed that although much manual testing is done, testing training is lacking (61% had little or no training, against 18% who claimed considerable). Most of the training was on-the-job or self-learning. Software tester skills may be improved by more formal courses on the job, through tertiary institutions, or vocational training. Training from external organisations or University courses was relatively small at 23% and 18% respectively

## 4.3 Testing Tools

The survey did not examine all the different type of tools, but automated testing tools (which relate to test execution and playback) and some of the support tools (configuration management, change control and problem tracking tools) were examined. The results showed that 34 (56%) of the organisations did not use automated software testing tools. One possible reason for this could be the high learning curve associated with automated testing tools. Also among those that actually used automated testing tools, eighteen said they used in-house built testing tools possibly due to the fact that the cost of automated testing tools on the market was very high. Nineteen specified that they used one or more tools, of which Rational was the most common. However, automated testing tools were considered beneficial as 62% of those that used automated tools claimed that they contributed to improving the quality of their software.

Among the support tools that were available (configuration management, change control and problem tracking tools), problem tracking and change control were used commonly. Configuration management is the overall support tool that keeps track of all the changes in the software development and coordinates the process (Kit 1998, pp.42-43). However many of the organisations adopted certain parts of configuration management (e.g. change control) as the support tool. Seven respondents did not use any support tools and this could either mean a very small project or poor set of procedures.

## 4.4 Testing Standards

It has been mentioned that standards provide commonality of communication and consistency within procedures (Kit 1998, p.44). However, there is currently no single common standard that governs software testing. The survey asked respondents whether they were aware of some of the popular standards. Many of the organisations were not aware of either IEEE or BCS standards (59% and 75% respectively). 51% of the organisations did not follow any standards at all. For those that followed software testing standards, 24 (41%) of them followed an internally developed standard rather than the standards setup by large software governing institutes. This may indicate either that the existing standards do not fit well the current needs of the organisations or that New Zealand organisations are just not aware of these standards. This calls for improvement in existing standards and awareness of testing standards in general.

### 4.4.1 What to Test

This is well covered in IEEE and BCS scheme standards, yet these were not readily used by software organisations in New Zealand. The internally developed standards will also cover specifically what is required for testing, but 51% of organisations did not follow any standards. Organisations may be able to cope without standards when there is only small number of employees, but as the organisations grow, clear standards are necessary.

### 4.4.2 Level of Testing

The level of testing can usually be expressed using CMM. However, because not all organisations follow this model, a question was asked in the survey about the portion of testing relative to the whole software development process. The median for this question was between 20-29% (the mode was 10-19% with only one organisation reporting over 60%). By taking into account the whole software development lifecycle, 20-29% spent in testing may appear appropriate. However this depends on the situation and it is difficult to evaluate this figure.

It does give an indication on the level of testing that is carried out by New Zealand software organisations and many were satisfied with their testing process. Only 8 (13%) responded that their testing process was 'Very Poor" or "Poor".

### 4.4.3 How to Test

Although some existing standards cover different techniques and methods that are available, they do not specifically cover how software should be tested. Therefore, software organisations themselves should have appropriate procedures which are well documented, so that new employees can refer to them easily. 58% of the respondents claimed that they document procedures. Although this may appear far from ideal, it may be acceptable in small organisations where there are only few employees and they know each other's tasks. However ideally, standards including the correct procedures should all be well documented.

## 4.5 Test Management

Coordination of the software testing process must be done effectively in order to achieve successful testing and software products. Test cases should be well planned and support tools such as configuration management should aid the overall process. The staff should have appropriate experience and all the documentation should be up to date for an efficient testing process.

### 4.5.1 Planning and Design

Test cases should be planned before they are written/recorded to avoid unnecessary duplication and to maximise benefits. However, only about 10% of the organisations surveyed indicated that they wrote test plans for all of their cases. 23% of the participants claimed they did not write any test plans. Writing test plans can be time consuming, and some business may be able to cope without writing test plans if they are relatively small. This indicates lack of good test management.

### 4.5.2 Configuration Management

Configuration management is divided into four main categories; configuration identification, configuration and change control, configuration status accounting, and configuration audit. Configuration management is one of the software testing support tools. Another common support tool is problem management (often called problem tracking tools) (Kit 1998, p.151). The majority of the organisations used change control (68%), one aspect of configuration management. However, problem tracking was the most popular support tool with 72% of organisations using this particular tool. New Zealand organisations, that use problem tracking and change control tools, may not be big enough to fully implement configuration management or problem management. Therefore they appeared to utilise certain parts that were most relevant in their testing process. 12% of the respondents did not use any support tools.

### 4.5.3 Testing Training and Experience

It appeared that the testing role in New Zealand was not clearly defined and not well recognised. More than half of the respondents said that they or their organisations received little or no training in testing and many of the organisations relied on developers to carry out testing. However, testing require skills that are different from programming (Dustin, Rashka *et al.* 1999, p.23). Ideally testing should be done by separate personnel. A counter argument to this could be that many organisations in New Zealand employ less than five people and therefore may not have enough resources to invest separately in testing. In small organisations developers were primarily responsible (82%) for the testing. It seemed that the people in these small organisations needed to be multi-skilled in order to survive. The majority of their experience in testing came from on the job or self study rather than specific courses provided by tertiary educators or external IT training organisations. Courses devoted solely to testing were difficult to find among New Zealand Universities as more emphasis is put on development rather than testing. For those organisations that had separate testing roles, 46% of the respondents stayed in their testing role for less than 3 years. However, 39% did not even have a separate testing role, as the organisations were very small.

### 4.5.4 Documentation

Documentation is important in any part of software development and testing is no exception. However, many developers and testers tend to put off documenting work (Kit 1998, p.48). Overall 37% of the organisations did not document their procedures or standards (this increased to 56% in small organisations). Good documentation reduces unnecessary communication and coordinates the process. Clearly documented standards and procedures allow new staff to settle in quickly. Also as organisations get bigger, good documentation is essential. However, due to small size of New Zealand software organisations, documentation was often overlooked. It did appear though, as organisations got bigger and employed more testing staff, the level of documentation increased.

### 4.5.5 Review

Reviews are necessary not only in software testing, but in any environment in order to evaluate existing processes/products and make continuous improvements (Perry 2000, p.66). Reviews in software testing involve inspection or walkthroughs where people, process or product may be reviewed. 25% organisations responded that they carried out inspections and 21% carried out "formal methods." As mentioned in section 4.4.2, the terms, such as "formal methods", may have been misunderstood by some respondents and therefore the results should be interpreted with caution. Reviews such as walkthroughs usually involve at least two people and can be quite time consuming (Lewis 2000, p.54) and therefore small organisations in New Zealand may lack resources to carry out structured reviews.

## 4.6 Testing Metrics

Metrics are useful for indicating status of software or process of software testing. Metrics provide simple quantifiable results that are easily understood. Metrics can indicate if software quality is improving or measure the severity of faults discovered during the testing process. Metrics can provide traceable evidence on the performance or quality of the testing process (Kit 1998, p.50). Overall, only 50% of the organisations commonly used metrics.

### 4.6.1 Performance Measure

Performance measures include test coverage and measuring efficiency. Some of the metrics used were to measure faults or problems (34%) and measure test case status and results (34%) or coverage (16%), while 15% said they used metrics to measure efficiency (E.g. Time taken in responding to a fault). One responded that customer satisfaction was the only metric that mattered. Although this may be true in some sense, performance metrics are important for self evaluation and continuous improvement. Overall, organisations in New Zealand often overlooked the importance of software testing metrics.

### 4.6.2 Quality of Software

Metrics which indicate quality of software include number of faults/problems, results/status of test cases, and other measures such as cyclomatic complexity. These measures are useful in evaluating the quality of software and important in making the decision as to when to ship the software. New Zealand organisations may cope without metrics when the firm is small and there are only few employees. However as an organisation grows, metrics become important when reporting to senior managers as they indicate performance and quality in a simple quantifiable manner (Kit 1998, p.51). Pressman (2005, p.649) also stated that by measuring, you can make objective evaluations and make improvements. However 40% of the organisations that used testing metrics either did not think or were unsure if they improved quality.

### 4.6.3 Criteria to Stop Testing

Deciding when to stop testing is probably one of the hardest questions in software testing. This answer varies greatly among organisations and there were many 'other' responses for this question. Metrics such as percentage of test cases passed and percentage of requirements met were popular when determining criteria to stop testing. 21% of organisations did not have any criteria and 20% stopped testing, because the project deadline was reached. Also some of the 'other' answers included vague answers such as 'when it works', 'when product appears to be bug free' and 'when it all works.' This may indicate lack of clear procedures within software organisations in New Zealand; they still operate in an "*ad hoc*" manner. This may be blamed on the small nature of New Zealand organisations, but it indicates that testing is not given the important status it deserves within software development.

## 5 Discussion

Perceived quality of software was treated as the dependent variable, and various hypotheses were tested using different aspects of testing as independent variables. Due to inadequate sample size and no *a priori* postulated relationship between perceived quality and actual quality, the results are not confirmative. Although there have been studies linking perceived behaviour with actual behaviour (Ajzen 1991), the same cannot be said between perceived quality and actual quality. Also perceived quality from the developers is likely to be biased. Nevertheless, hypothesis testing carried out in this study gives some indication and can be taken into account for future research. The only statistically significant relationship from hypothesis testing was between perceived quality and number of employees. It was surprising to find that perceived quality of their software from smaller organisations was higher than the larger organisations. This may indicate that small team of testing staff are more involved in the development of software and have greater pride in their work, leading to higher perceived quality. Perhaps the sole person who is in charge of testing may perceive quality is good. It may also be the fact that smaller organisations may be able to cater to customer needs better and actually produce better quality. This relationship should be researched more in depth and analysed further. Although five hypotheses were unable to be analysed statistically, the cluster report does give an indication that adoption of support tools and metrics are likely to lead to higher perceived software quality (Sung, 2005).

Overall, despite the lack of some key components in software testing, the majority of the respondents were satisfied with their testing process. It is difficult to say whether they were satisfied given their circumstances or whether they believed their testing was done at a satisfactory level. Only 13% of the respondents said their process was below satisfactory. This may contradict results found in other parts of the survey and this is open for more research. However, some of the comments indicate that respondents are satisfied with their testing process and product taking into account their limited resources. Those that were satisfied with the product and process still mentioned that there was room for improvement. It is difficult to judge whether this is acceptable as New Zealand organisations are not fully meeting all the requirements of software testing. However, given their limited resources, they may be performing to the best of their ability.

## 5.1 Comparison with Australia Testing Survey

Many of the findings are similar across the Tasman. A large fraction of the organisations in New Zealand and Australia were still performing "*ad hoc*" testing without clear methodologies and use of testing metrics. The reason for this in New Zealand appears to be small sized organisations with a small number of employees. Employees need to carry out a wide variety of tasks and cannot be just restricted to testing. Therefore with a smaller number of employees, time and money were the two main barriers in adopting a methodology for New Zealand organisations. However in Australia, lack of experience was a key factor for organisations still carrying out "*ad hoc*" testing (Ng, Murnane *et al.* 2004). There was also lack of software testing courses at tertiary institutions. However there seemed to be more courses from external IT institutions in Australia (Ng, Murnane *et al.* 2004) than in New Zealand.

A large number of organisations on both sides of the Tasman did not follow published software testing standards. They both relied on internally developed standards and many did not even follow any standard. Standards need to be generic enough so that they can be applied across most of the organisations, yet specific enough so that they can be used as guidelines to be followed. The purpose of Sung's (2005) study was not to propose set of standards, but to develop an underlying framework from which the standards could be derived. Here we evaluate the status of testing in New Zeland using Sung's (2005) framework (Table 3).

By covering the key aspects of software testing and understanding the nature of software testing in New Zealand, the proposed framework can be further refined and be used to create  and evaluate a common set of testing standards.

Automated testing tools were more common among Australian organisations than New Zealand ones. This is probably reflected by the bigger market and bigger organisations in Australia. Therefore those organisations would have had more money and resources to spend on automated testing tools. Also use of external testers was more common in Australia with 37% of the participants having used external testers (Ng, Murnane *et al.* 2004). Only 8% of participants in New Zealand claimed that they outsourced the testing process to a third party. Perhaps there are more businesses specialised solely for testing in Australia than in New Zealand. The Australian survey found that use of structured methodology was lower amongst government organisations, but there was no evidence of this in New Zealand. In fact, government organisations in New Zealand were generally bigger than private firms, so they had more resources allocated towards software testing.

## 6 Conclusion

Software testing is becoming more important in software development and its importance is being recognised across the industry. However, clear common guidelines and standards that govern software testing are still lacking. Sung (2005) integrated some of the major existing software testing standards. The framework created was used to develop a survey of the status of software testing in New Zealand. The survey indicated that due to the small nature of many software development organisations in New Zealand, many of the key aspects were often overlooked and testing did not gain the recognition it deserved. This was somewhat inevitable in many small organisations where employees had to be multi-skilled and testing role could not be separated.

**Acknowledgements:** The survey was run on software provided as an educational grant by websurveyor. The survey server was purchased from a University of Auckland Research Committee (AURC) grant.

**Table3: Summary of the key findings from survey against Sung's (2005) framework**

| Testing Framework | Comments |
|---|---|
| *Testing Methods and Techniques* | Testing methods such as manual testing and black box testing were popular among New Zealand organisations, possibly due to fewer staff and lack of resources. Methodologies that require more time and experience such as inspection reviews were not as popular. |
| *Testing Tools* | Testing tools were also not used as commonly as expected. Over half of the respondents did not use automated testing tools and some of the support tools were also overlooked by some organisations. Tools were most popular during the test execution, reporting and evaluation phase of testing. |
| *Testing Standards* | There was a lack of adoption and also awareness of some of the major software testing standards. This also indicates the need for a common standard that will be adopted by most practitioners. |
| *Test Management* | Overall, test management among New Zealand organisations was rather weak. Testing staff did not appear to receive adequate training and documentation. Test plans were overlooked by some organisations. |
| *Testing Metrics* | Testing metrics were also not used frequently by New Zealand organisations. Despite the importance of the metrics stated in textbooks, many organisations did not put this into practice. Possible cause could be small testing team and lack of experience. |

## 7 References

Ajzen, I. (1991). "The theory of planned behaviour." Organ. Behaviour and Human Decision Processes **50**(2): 179-211.

Buckley, F. J. (1984). The IEEE Software Engineering Standards Process. ACM'84 Annual Conference.

Chakraborty, A. K. and T. S. Arthanari (1994). "Optimum Testing Time for Software under an Exploration Model." Operation Research Society of India **31**(3): 202.

Dustin, E., J. Rashka, *et al*. (1999). Automated software testing: introduction, management, and performance, Addison-Wesley.

Fournier, R. (1998). "Testing, testing... a rigorous process." InformationWeek(679): 8A.

Frazer, L. and Lawley, M. (2000), Questionnaire design and administration - a practical guide, Wiley.

Hendrick, K. E. (1999). IBM Rational Robot: Closing the Quality Gap with Rational Suite TestStudio.

Kit, E. (1998). Software Testing in the real world. New York, Addison-Wesley.

Lee, J. Y. T. (2002). Effective Web Site Design and Usability for E-Tailers in New Zealand. Department of Information Systems and Operations Management. Auckland, Unpublished MCom Thesis, University of Auckland.

Lewis, W. E. (2000). Software testing and Continuous Quality Improvement. Florida, Auerbach.

S. Ng, T. Murnane, K. Reed, D. Grant and T. Y. Chen, A Preliminary Survey on Software Testing Practices in Australia, Proceedings of the 2004 Australian Software Engineering Conference (ASWEC 2004), 116-125, 2004.

Paredes, D. (2005). New Zealand's 100 biggest users of IT. Managing Information Strategies. **8**.

Perry, W. E. (2000). Effective Methods for Software Testing. New York, Wiley.

Pinsonneault, A. and K. L. Kraemer (1993). "Survey Research Methodology in Management Information Systems: An Assessment." Journal of Management Information Systems **10**(2): 75-105.

Pressman, R. S. (2005). Software Engineering - A Practitioner's Approach. Singapore, McGraw-Hill.

Reid, S. (2000). BS 7925-2: the software component testing standard. Quality Software, 2000. Proceedings. First Asia-Pacific Conference on, Hong Kong.

Reid, S. (2000). Software Testing Standards - do they know what they're talking about? Retrieved 28 September, 2005, from http://www.testingstandards.co.uk/publications.htm

Schach, S. R. (2002). Classical and Object-Oriented Software Engineering. With UML and Java, McGraw Hill.

Schwartz, K. D. (2003). "Tools automated software testing." InformationWeek(955): 55.

Software Engineering Technical Committee (1983). IEEE Standard for Software Test Documentation, Institute of Electrical and Electronic Engineers Computer Society.

Software New Zealand (2005). Software New Zealand. 2005. Retrieved 8 September, 2005, from http://www.nzsa.org.nz/

Sung, P. W-B. (2005) A Framework of Software Testing Standard and Survey on Software Testing Practices in New Zealand. Unpublished BCom (Hons) Research Essay, University of Auckland.