Mitigating the Impact of Software Test Constraints on Software Testing Effectiveness

Grafton Whyte and Donovan Lindsay Mulder
University of the Western Cape, Cape Town, South Africa
drgwhyte@aol.com
donovanmulder@hotmail.com

Abstract: Software testing is the one of the primary methods used in the validation and verification of output in the software development industry. It is seen as a key method for achieving software quality, reliability, fitness for purpose and customer satisfaction. Software testing is however an expensive process accounting for as much as 50% of the cost of developing software based systems. In recent years, software testing as a discipline has come under pressure due to time, cost and skills constraints. These constraints impact negatively upon software test effectiveness. Therefore it is critical to identify and implement test tools that reduce the negative impact of software test constraints on software test effectiveness. In this paper the researchers examines some of the most popular software testing tools such as test case prioritisation, test suite reduction and test selection criteria, to identify: Which *individual* test tools are most likely to yield optimal test effectiveness and, Which *combination* of test tools is most likely to yield optimal test effectiveness and mitigate the effect of test constraints An extensive review of the software testing literature was conducted and used to construct a survey instrument as the basis for examining the impact of test constraints on software test methodology. The survey was issued to expert software test practitioners from various locations globally; the sample consisted of 43 test cases. The main findings were that no one approach to testing would yield satisfactory results but a combination of two or more test types from Automated testing, Smoke testing, Test case prioritisation and Regression test selection could yield effective software testing results and mitigate the effects of test constraints.

Keywords: software test tools, software test effectiveness, software test constraints, test selection methodology, test case selection criteria

1. Introduction

As the significance of software increases aspects such as quality, reliability and customer satisfaction have become strategic factors for software development organisations (Huang, 2005). There are various methods of software verification and validation. These are reviews, walkthroughs, software inspections, formal methods and software testing. Of these, software testing has been the method of choice for software validation and verification. Software testing is a costly and unavoidable task (Bertolino, 2007). It is a complex and arduous task (Shahamiri, 2009; McMinn, 2009; Srikanth et al., 2005a & b) which can consume more than 50% of the cost of a software development project without adding any basic functionality to the end product. It however, remains the method of choice through which confidence in the end product is realised (Ramler and Wolfmaier, 2006). Engel and Last (2007), state that inadequate execution of software and systems verification, validation and testing account for losses that can eclipse more than 10% of company's turnover.

In today's software development environment testing has come under pressure due to shorter product time to market, shrinking budgets and higher quality demands (Ramler and Wolfmaier, 2006; Srikanth et al., 2005). According to Bryce and Colbourn (2005) software testing is an expensive time consuming process which is often restricted by cost and time constraints. According to a study by Huang (2005), defect detection rate is impacted by the skill level of test personnel and size of the project. As does testability (Berner et al., 2005) of the system. Do et al., (2008) argue that time constraints impact upon regression testing and negatively impacts on the cost effectiveness of test case prioritization methodologies. According to Berner et al., (2005) in most instances, systems are hard to test because of the "cumbersome architecture" rather than the complexity of the system. Often this is due to poorly specified requirements with inadequate description of user feedback. This is particularly apparent in test automation where test cases rely on user feedback in order to execute as expected.

Software testing drives are commonly beleaguered by constraints such as time, cost, and insufficient skills. These constraints impose risk on the realisation of software test effectiveness with respect to software testing goals. Understanding how to mitigate this risk is a key-factor in achieving successful software testing. **This research aims to identify software test tools which can increase software**

test effectiveness and thereby support effective software testing in the presence of software test constraints.

In order to realise this objective the following research questions are answered.

Main Research Question

- What test approaches are most likely to reduce the negative impact of test constraints on test effectiveness?

Sub-Questions

- What is software testing?
- What is software test effectiveness?
- What are the goals of software testing?
- What is effective software testing?
- What are the existing software test constraints and what impact do these constraints have on software test effectiveness?
- What are the existing software test tools and what impact do these tools have on software test effectiveness?
- What are the existing test selection criteria and what impact do these criteria have on software test effectiveness?

This paper is organised as follows: A literature review is presented in section 2, the research methodology is presented in section 3, the research results are presented in section 4, a discussion of the results and the test management implications is presented in section 5 and a conclusion with recommendations for future research is presented in section 6.

2. Literature review

2.1 Terminology

For the purposes of this paper the researchers define a:

- **Test tool** as any methodology, process or know-how that contributes to software testing and as such is not limited to software based test tools.
- **Test approach** as one or a combination of test tools used to implement software testing.
- **Test constraint** as any factor that inhibits the software testing process from achieving the desired levels of performance.
- **Test Design** refers to test case content and test case size.

2.2 Effective software testing

In the following discussion the research sub-questions 2 to 6 are discussed. **Software testing** is defined as the observation of the execution of software based systems in order to verify that the system behaves as expected and to identify defects in the system under test (Bertolino, 2007). **Software test effectiveness** is defined as the number of defects found through software testing divided by the total number of defects (Vallespir and Herbert, 2009). Defects can occur at any stage in the software development lifecycle. Early identification of software defects is essential in order to minimise risk due to defect proliferation and minimise validation costs (Baresi and Pezze, 2006). **Goals of software** testing are improvement of software quality and reliability through defect detection resulting in increased customer satisfaction (Shahamiri et al., 2009; Huang, 2005; Baresi and Pezze, 2006). Risk mitigation is also an aim of software testing (Frank, 2000).

Given the statements above one can conclude that **effective software testing** possesses the following characteristics: (a) it is on schedule, (b) has high defect detection capability, (c) has high detection rate and is (d) cost effective.

- Software testing that is *on schedule* compared to software testing that is not, reduces the risk imposed by time constraints, such as product time-to-market, which is often critical for establishing or maintaining the company's competitive advantage.

- Software testing that has a high *defect detection capability* compared to software testing that does not, has a higher probability of finding hard to find defects early in the software development life cycle thereby assisting the company achieve positive customer perceptions.
- Software testing that has a high defect *detection rate* compared to software testing that does not, is more likely to find most defects early in the software development life cycle thereby contributing to the overall cost effectiveness of a software development drive.
- Software testing that is *cost effective* compared to software testing that is not is more likely to stay within budget. These characteristics are illustrated in figure 1.

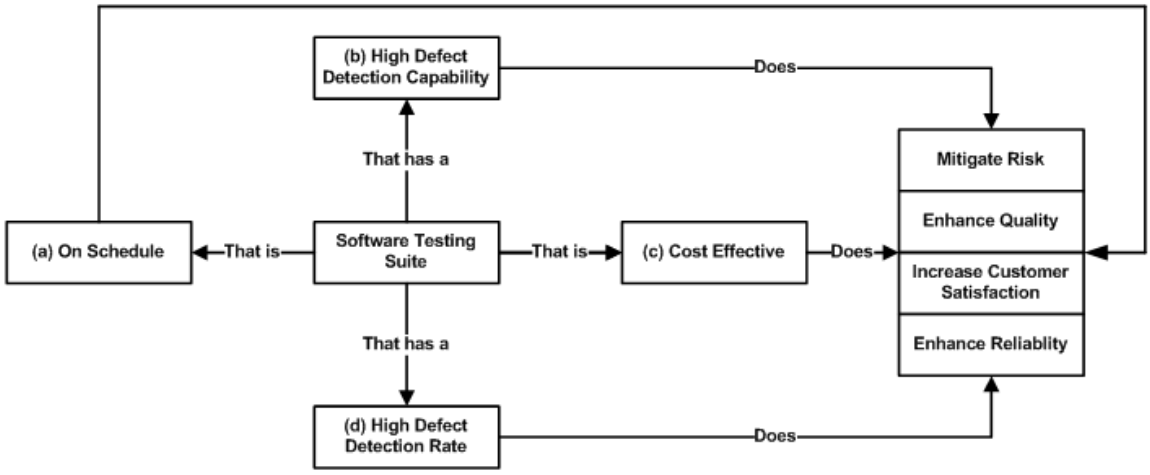

Figure 1: Characteristics and goals of effective software testing

The question that one has to ask here is, how are these characteristics of effective software testing realised in the presence of test constraints?

Bryce and Colbourn (2005) state that due to time and cost constraints entire test suites in most instances are not run. In these instances it is of utmost importance to prioritise tests. Test case identification and prioritisation models are methods for reducing the cost and increasing the effectiveness of software testing through test case relevance modelling and prioritization (Rothermel et al., 2004). Rothermel et al., (2004) discusses four regression test methodologies from a test design perspective and concludes that test design plays a critical role in defect detection capability and rate. Automated testing is another method that promises greater testing coverage in shorter test cycles (Shahamiri et al., 2009) and has been proposed as a method to minimise costs (Ramler and Wolfmaier, 2006). Bertolino (2007) states that test automation is critical to "cost-effective test engineering." Factors such as test design (Rothermel et al., 2004) and test oracles (Memon and Xie, 2005) are presented as means of increasing the effectiveness of software testing. Smoke testing is presented as an example of a test approach used to detect defects early in the software development lifecycle (Memon et al., 2003).

The following discussion addresses the research sub-questions 7 and 8 by reviewing test design with respect to regression test selection methods; two test selection methods (test case prioritisation and test suite reduction); test oracles and automated testing are discussed. Smoke testing is presented as an example of a software testing approach that finds a significant amount of defects early in the software development life cycle.

2.3 Test design

Rothermel et al., (2004) discusses four regression test methodologies from a test design perspective. Under experimental conditions two design factors are considered:

- **Test suite granularity** which is pertinent to the test case size (granularity) and pertains to the applied input count per test case.
- **Test input grouping** which is pertinent to test case content and pertains to the "heterogeneity or homogeneity" between test case inputs.

Two null hypotheses were used to evaluate the impact of test design on the regression test methodologies. These are:

- H1: “test suite granularity does not have a significant impact on the costs and benefits of regression testing techniques”
- H2: “test input grouping does not have a significant impact on the costs and benefits of regression testing techniques”

H1 was rejected while H2 could not be completely rejected (only in two cases it could be partially rejected).

The four methodologies investigated were “retest-all, regression test selection, test case prioritisation, and test suite reduction”. All four were shown to have increased cost effectiveness, defect detection capability and reduced test execution time with the application of relevant test case design. Rothermel et al., (2004) highlights the cost-benefit trade-offs associated with test design with respect to these regression test selection methodologies. Granularity has a greater impact on “retest-all, regression test selection and test case prioritisation”; but a less significant impact on “test suite reduction”. Test input grouping had a greater impact on “test suite reduction” and less significant impact on “retest-all, regression test selection, test case prioritisation”.

Coarse granularity compared with fine granularity test suites have a greater defect detection capability on easy to detect defects. Whereas fine granularity test suites are more capable of revealing hard to detect defects when compared to coarse granularity test suites. This capability of coarse granularity is attributed to the fact that the probability of exercising functionality that induces data state and output change is greater. Fine granularity test suites can better support selection and prioritisation with a resultant effect of reduced test execution time, increased cost effectiveness and high defect detection rate of hard-to-find defects. This is attributed to the support fine granularity affords test selection given that tests can be selected against certain criteria to achieve specific goals. See table 1.

It is clear from this section that test design has a significant impact upon effective software testing as defined in section 2.2 and figure 2 is a basic diagram that illustrates this relationship. Rothermel et al., (2004) concludes that building flexibility into test suites which affords readjustment of granularity to address test effectiveness at any stage of testing is critical.

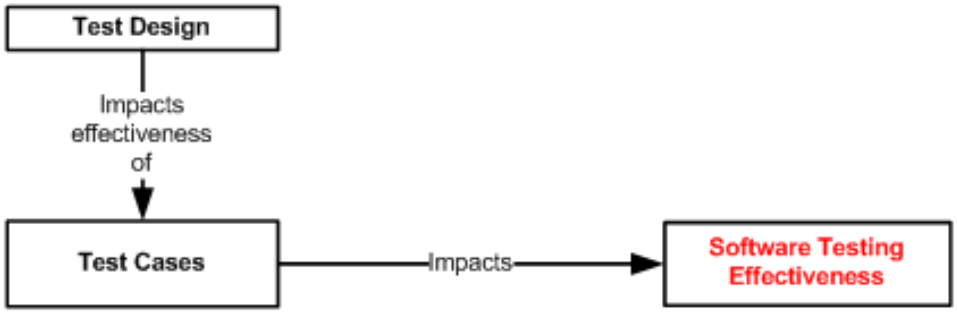

Figure 2: Test design and impacting software testing

A question here is what else can affect the effectiveness of test cases? Rothermel et al., (2004) mentions that test oracles play an important role in the cost effectiveness and the defect detection capability of test cases. A short discussion on test oracles is provided in the next section.

2.4 Test oracles

Test oracles are defined as an accepted dependable source of specified input and expected output of software behaviour and a means of reconciling expected and actual behaviour (Shahamiri et al., 2009). Test oracles have a significant impact on the defect detection capability and cost effectiveness of a test suite (Memon et al., 2003; Memon and Xie, 2004, 2005). Small test cases impose the risk of weak defect detection capability on test suites. A strong test oracle counteracts this risk by enhancing defect detection capability (Rothermel et al., 2004).

Table 1: Impact of granularity on regression test methodologies: RTA, RTS, TSR and TCP

Regression Test Methodology	Fine Granularity	Qualification	Coarse Granularity	Qualification
Retest-All (RTA)	Increased Test Execution Time	Greater number of tests to setup test setup time	Decreased Test Execution Time	Less test setup time
	Decreased Cost Effectiveness	Greater cumulative cost of test cases	Increased Cost Effectiveness	Less test cases therefore cumulative test case costs are reduced
	Diminished Defect Detection Capability	Small test cases do not test many data and program state changes	Enhanced Defect Detection Capability	Large test cases possess a higher probability of encountering data and program state changes
Regression Test Selection (RTS)	Decreased Test Execution Time	Greater test selection flexibility	Increased Test Execution Time	Less tests to select from therefore negates opportunities to choose efficient test case combination
	Increased Cost Effectiveness	More tests to select from therefore the most cost effective combination can be selected	Decreased Cost Effectiveness	Less tests to select from therefore negates opportunities to choose cost effective test case combinations
	Diminished Defect Detection Capability	Small test cases do not test many data and program state changes	Enhanced Defect Detection Capability	Large test cases possess a higher probability of encountering data and program state changes
Test Suite Reduction (TSR)	Enhanced Defect Detection Capability of hard-to-find defects	Greater test reduction flexibility	Decreased Defect Detection Capability of hard-to-find defects	Fewer opportunities for including test cases in which fault-masking interactions do not occur
Test Case Prioritisation (TCP)	Decreased test execution time at the end of the development cycle	Greater test selection flexibility	Decreased test execution time	Less test setup time
	Enhanced cost effectiveness	More tests to select from therefore the most cost effective combination can be selected	Increased cost effectiveness	Less test cases therefore cumulative test case costs are reduced
	Enhanced Defect Detection Capability of hard-to-find defects	Tests are selected to achieve specific goals and with desirable characteristics such as a high defect detection potential	Enhanced Defect Detection Capability of easy-to-find defects	During this phase test execution is expected to defect easy-to-find defects

However strong test oracles increases test execution time and reduces cost effectiveness (Memon and Qing, 2005). A weak test oracle could result in reduced test execution time though this might be due to misleading or incomplete oracle information. There are risks of defects not being detected. Memon and Qing (2005) conclude that test cases lose their defect detection capability substantially, through decrepit test oracles. Comprehensive test oracles employed at the end of the execution of a test case yields the best cost benefit ratio. Rothermel et al., (2004) state that coarse grained test

cases minimise the impact of weak test oracles on defect detection capability. Thus test design plays a significant role in increasing software testing effectiveness when employing test suite reduction.

Test case content is an integral component of test oracles and is commonly used to determine the suitability of a test case against specified test selection criteria to realise specific testing goals. Test oracles play a critical role in software testing effectiveness as defined in section 2.2. This relationship is illustrated in figure 3.

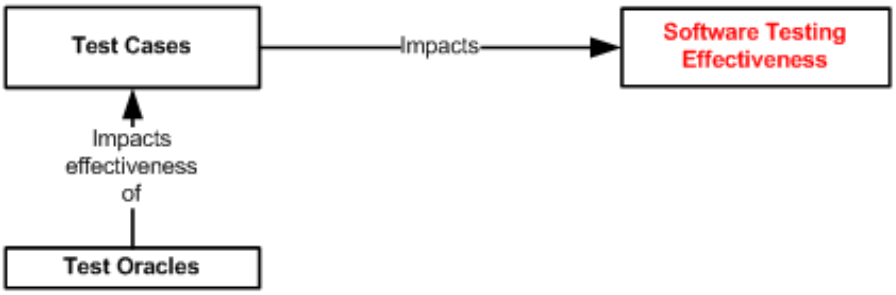

Figure 3: Test oracle impacting software testing

In the next section a short discussion on test selection criteria with respect to Test Case Prioritisation and Test Suite Reduction is presented.

2.5 Test selections methods and test selection criteria

Sampath et al., (2008) state that test case prioritization techniques minimise the impact of time constraints. Test suite prioritization techniques are shown to enhance the defect detection rate early in the development cycle compared to random test selection (Rothermel et al., 2004; Srikanth et al., 2005a). Test cases are ordered according to some explicit criteria designed to expose defects as quickly as possible. These criteria could be code coverage, possibility of defect existence and defect detection potential (Sampath et al., 2008; Do, et al., 2008).

Srikanth et al., (2005b) states that approximately 50% of all defects are generated in the requirements phase. In the study Srikanth et al., (2005a) assert that tests are prioritised according to requirements in order to detect high risk defects quickly. Prioritisations of requirements are based on four factors: “requirement volatility, customer priority, implementation complexity and fault proneness”. It is concluded that compared with random ordering of tests cases, prioritisation of requirements based testing increases test effectiveness which significantly contributes to the increased defect detection rate of high risk defects.

Test suite reduction aims to remove nonessential test cases permanently while keeping the most effective test cases. The goal of test suite reduction is to reduce the cost of regression testing (Parse et al., 2009) by satisfying all test requirements with the least amount of test cases (Zhang et al., 2008a, b). Sampath et al., (2008); McMaster and Memon (2008) state that test suite reduction methodologies are based on test criterion which reduces the size of the test suite without reducing use case delineation and defect detection effectiveness. Code coverage, functional coverage and defect detection capability are commonly used as test reduction criteria (Parse et al., 2009). This leads to reduced cost and time of test execution and test suite management (Rothermel et al., 2004). Zhang et al., (2008b) states that cost effective test suite reduction can be achieved through the optimisation of test requirements as this leads to smaller test suites. Some studies have shown that test suite reduction can significantly increase cost effectiveness of a test suite with minimal loss in defect detection capability, while others have shown that test suite reduction can significantly reduce the defect defection capability of reduced test suites. Discarding test cases can quickly result in a significant decrease in the defect detection capability of the reduced test suites.

In the cases of Retest-All and Regression Test Selection as Test Selection methods, Retest-All always as the name indicates, selects all test cases; Regression Test Selection methods assume that test cases that do not test changed functionality will not detect defects (Engström, 2010), therefore the the test selection criteria would be ‘select all test cases that test changed functionality’.

In conclusion test selection criteria impact the effectiveness of test selection methods from the perspective of cost, time, defect detection rate and capability. Test selection methods in turn impacts upon software testing effectiveness as defined in section 2.2. Test cases impact software testing effectiveness through its relationship with test selection criteria from the perspective of test case relevance. This relationship is illustrated in figure 4.

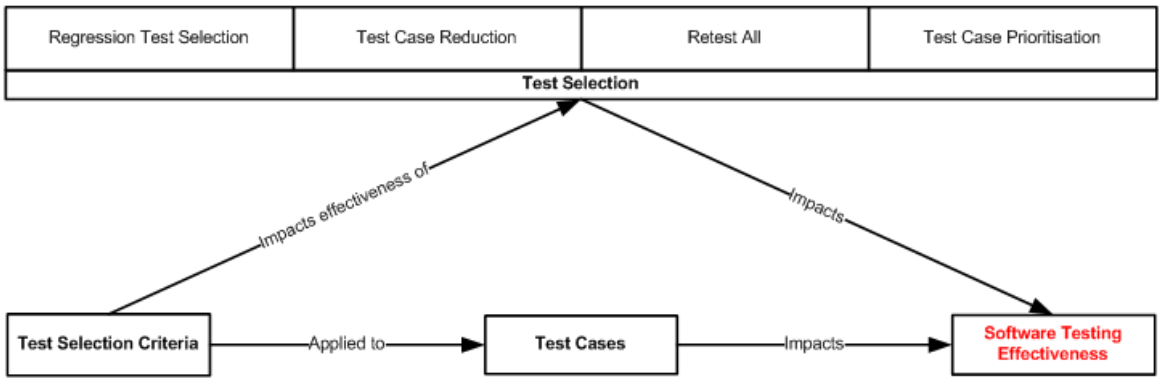

Figure 4: Test selection criteria impacting test selection, test cases and software testing effectiveness

So far we have considered test suite construction methodologies (test tools). However, these test tools are not exclusive to manual testing. They are also used in automated test suite construction. In the following section we discuss automated testing from a test coverage and test execution rate perspective. Smoke testing is also discussed as a quick and cost effective approach for achieving rapid defect detection early in the development cycle. Both are considered to be test tools.

2.6 Automated testing

Karhu et al., (2009) states that automated software testing is the process of automating tasks that comprise software testing. These tasks are processes such as test data generation; test script development and execution; verification and validation of test requirements and the implementation of test automation tools.

Shahamiri et al., (2009) argues that test automation has been one method used to decrease the costs of software testing. This is supported by Ramler and Wolfmaier, (2006) and Karhu et al., (2009) who also states that the automated testing can be used in place of manual testing when time is a constraint. This is further supported by Zhu et al., (2006) who states that in order to reduce the costs and improve software testing effectiveness it is critical to automate the testing process.

Automation of regression testing is seen as a means of realising increased efficiency within the software testing process. Harman, (2008) states is it critical to automate the generation of test data in order to achieve cost effectiveness in software testing. However it has not been proven in research that it is possible to automate all oracle activities (Shahamiri et al., 2009).

Given that automated testing if implemented correctly speeds up test execution and defect detection rate, it will therefore impact upon software testing effectiveness as defined in section 2.2.

2.7 Smoke testing

Smoke testing is used to detect defects early in the software development lifecycle (Memon et al., 2003). It is widely accepted that early detection of defects leads to:

- Lower defect fixing costs
- Lower costs of formal testing
- Reduced cost of execution time of formal testing further down the line
- Enhanced software quality
- Risk minimization

Memon and Xie (2004) conclude that:

- For the majority of applications smoke tests are able to defect greater than 60% of defects.
- A substantial proportion of code can be tested by small (1 to 3 events) smoke tests.
- Large smoke tests with more events are able to detect more faults than small smoke tests.
- Smoke test effectiveness is significantly impacted upon by the test oracle.
- Application of a complete test oracle at the end or final event of a smoke test case yields the best balance of cost effectiveness and defect detection capability.

Smoke testing has the potential to dramatically improve the effectiveness of software testing as defined in section 2.2. Figure 5 illustrates the relationship between test selection criteria, test execution methods (of which smoke testing is an excellent example) and software testing effectiveness.

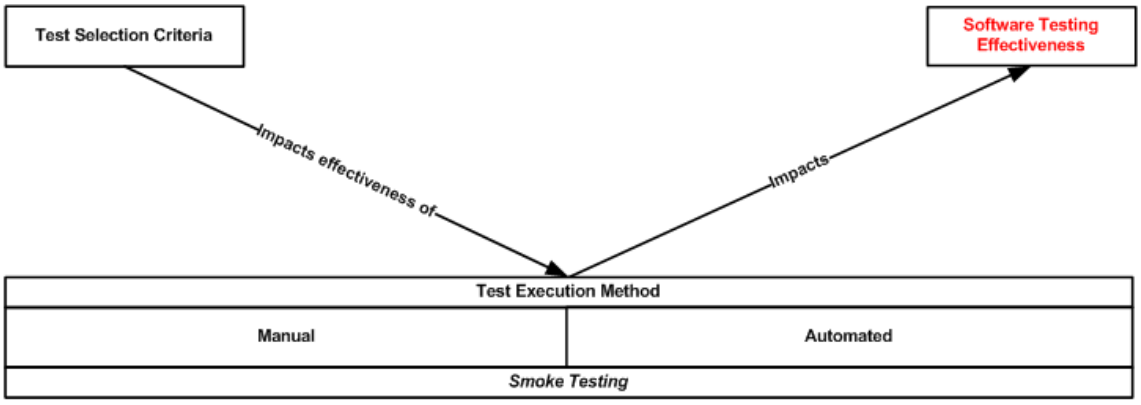

Figure 5: Test selection criteria impacting test execution and software testing effectiveness

2.8 Conclusion of the literature review

From the literature it is concluded that:

- The skill of human resources, project size, inadequate requirements, software testability, time, cost and test design are test constraints falling into the categories of time, cost and skills.
- Retest-all, regression test selection, test case prioritisation, test suite reduction, smoke testing, test automation, test oracles and test design are determined to be test tools used to minimise the impact test constraints on software testing effectiveness.
- Test selection criteria such as code coverage, possibility of defect existence, defect detection potential, test case design (test case content and size); requirement volatility, customer priority, implementation complexity and fault proneness are identified as test selection criteria which can support test case prioritisation, regression test selection and test suite reduction.
- These test tools and test selection criteria were found to directly impact test effectiveness through four aspects which featured prominently in the reviewed literature; these are (a) defect detection capability, (b) defect detection rate, (c) cost effectiveness and (d) test execution time.

The Software Test Effectiveness Model (figure 6) serves as a summary of the literature review. The model combines the sub models depicted in figures 2, 3, 4 and 5. This model summarises the relationship between Test Tools (Test Selection Criteria, Test Selection Methods, and Test Execution Methods) and Test Effectiveness as conceptualised for the purpose of this research. Derived from the ideas and concepts discussed in the literature review, this model provides a visual map of how all the test tools and test selection criteria fit together to form a software testing approach and how these factors collectively can enhance effective software testing.
The Software Test Tool model works as follows: the Test Selection Criteria impacts upon the effectiveness of the Test Selection Methods which in turn impacts effective software testing. Test Selection Criteria applied to Test Cases impacts upon effective software testing. Test Design and Test Oracles impacts upon the defect detection rate and capability of Test Cases which in turn impacts upon effective software testing. Test Selection Criteria impacts the effectiveness of Test Execution Methods which in turn impacts upon effective software testing.

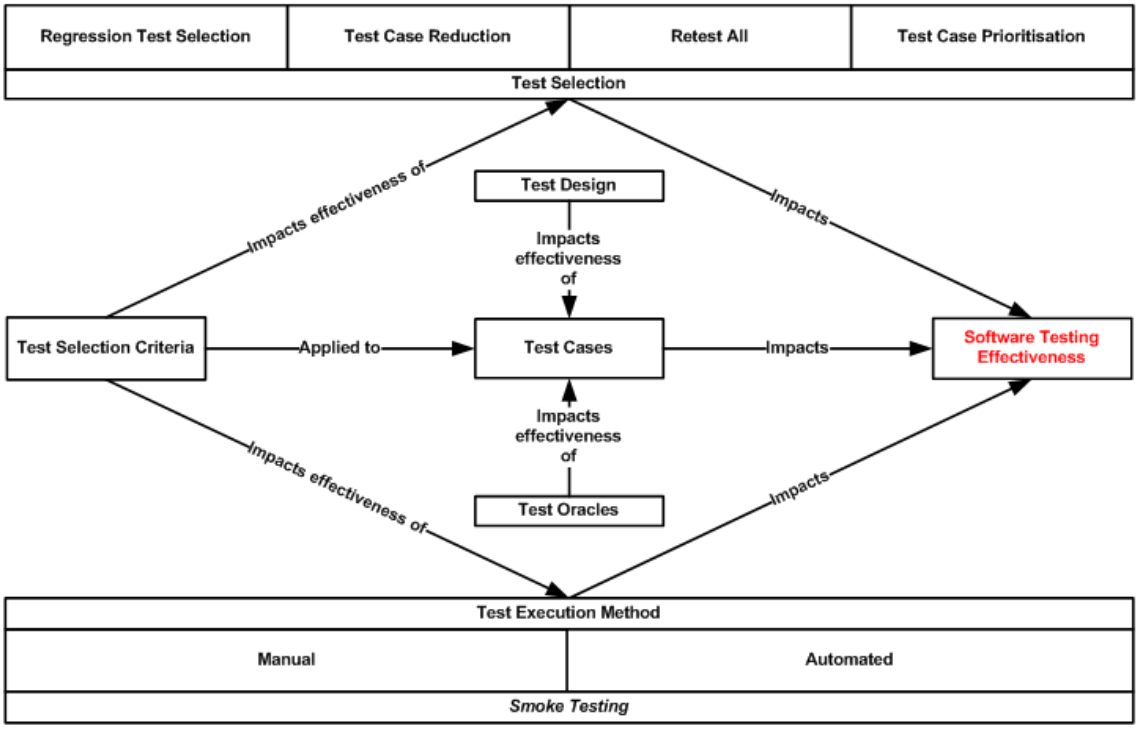

Figure 6: Software test effectiveness model (summary of the literature review)

The literature review answered the research sub-questions 2 to 8 from a theoretical perspective. In the following sections research sub questions 6, 7 and 8 are again answered, but from the perspective of software testing practitioners.

3. Research methodology

The ideas discussed in the literature review embody the Goal-Question Metric Approach template proposed by Basili et al., (1994): They propose analysing software testing tools, software test selection criteria and software test constraints for the purpose of knowing which software testing tools and test case selection criteria or a combination thereof increases software test effectiveness in the presence of software test constraints; with respect to their usage and possible software testing improvement potential from the view point of software testing professionals.

There are a plethora of test tools and test selection criteria in the current literature that can arguably minimise the negative impact of test constraints on test effectiveness. The usage of these test tools and test selection criteria differ widely, while information about the actual usage in the presence of test constraints in project situations seems to be missing. To assess the extent to which test tools minimise the negative impact, a research instrument was needed that could be measure the perceived impact of test tools on test effectiveness in the presence of test constraints. No such instrument was found in the literature. Most if not all of the research in the literature dealt with test tools used under experimental conditions and not in real projects. Also it is not certain how these test tools and test selection criteria would contribute to overall test effectiveness in live projects. Therefore a research instrument was developed based on test tools and test selection criteria found in the literature and that could be used to measure the perceived impact on actual projects.
A survey questionnaire consisting of closed questions was used as the research instrument. Quantitative data was collected. The research constructs are test tools, test case selection criteria, test constraints and test effectiveness. The survey was designed to elicit data from a practitioners' perspective. The aim was to identify the perceived impact (a) test tools have on test effectiveness, (b) test selection criteria on test effectiveness, (c) test constraints on test effectiveness and (d) the actual effectiveness of testing activities. The research instrument was piloted using 2 cases. The pilot study was interview based in order to gauge first-hand the effectiveness of the survey instrument. The instrument yielded coherent results with minor modifications. During the pilot it became evident that due to the wide range of experience held by the respondents each expert could provide data on 2, 3 or more projects. The question whether such a move would compromise the study was considered and rejected. Largely because the evidence suggested that software testing professionals tended to

select tests based on the criteria prevailing within a project environment, rather than just applying the same test tools for every project. This was also demonstrated by the range of tools and techniques that came under investigation in this research.

The research population is all software testing professionals in the world. However it is not assumed that statistical relevant results with high external validity will be obtained due to the fact that convenience sampling was used to determine the sample from the population.

One of the co-author (Donovan Mulder) throughout his career in software testing has made contact with many software testing professionals and is a recognised professional himself. These contacts were made mostly through software testing seminars, conferences and the professional environment. The contacts represent various industries such as software testing consultancies, professional services, software development, telecommunications, banking, finance, investment banking, retail and energy. Some have recent experience using the outsourced model, some have worked their way through the ranks from software tester to software test manager, and some are published authors of software testing books and peer reviewed articles. The questionnaire was sent to this sample.

Each respondent was asked to provide data for three projects. Each project was treated as a single case. In total 43 cases were reported on by the respondents. The survey questionnaire was sent to 18 members of the sample of which 2 did not respond in time giving a total of 16 responses (88.9%). Of the 47 cases received only 43 were deemed useable as the value of software test effectiveness was omitted in the 4 cases. Respondents were located in geographically dispersed locations (USA, England, Ireland, France, South Africa, India and Australia). The predominant roles were test managers and test leads (see figure 7). The case studies covered a range of industries with financial services and telecoms dominating (see figure 8).

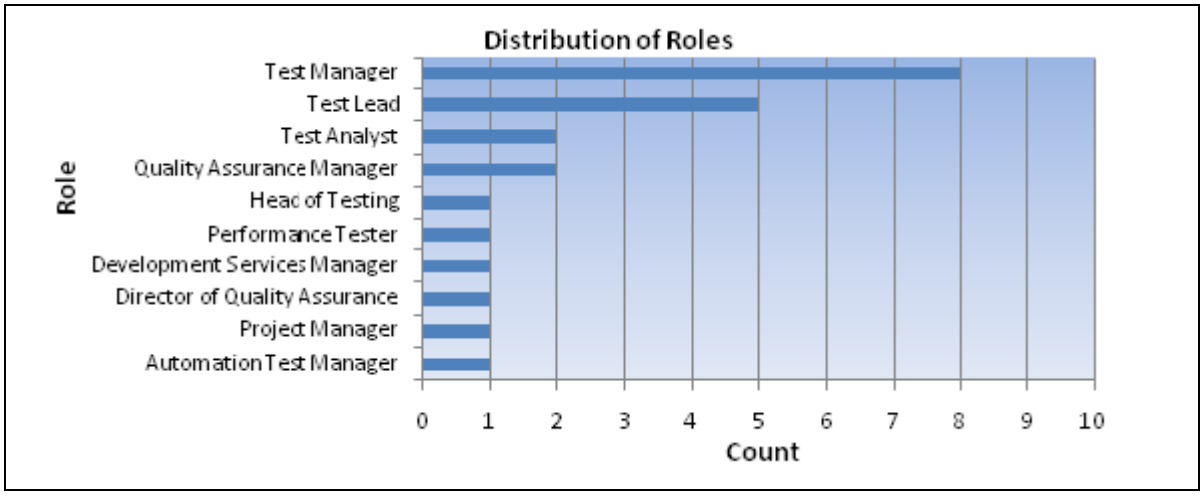

Figure 7: Distribution of roles

3.1 Validity

Construct Validity: Observations were based on the respondents experience and as such might have led to some differences in interpretation, to mitigate this effect definitions of all test variables were defined in the questionnaire.

External Validity: 43 cases (from 16 software testing professionals) were collected in this research. The results can be generalised to a certain extent as the cases are from different industries and from geographically dispersed locations (even from the same person as these professionals often operate internationally). Although convenience sampling was used to select respondents due to the dispersed nature of the sample it can be considered a fair representation of the professional population.

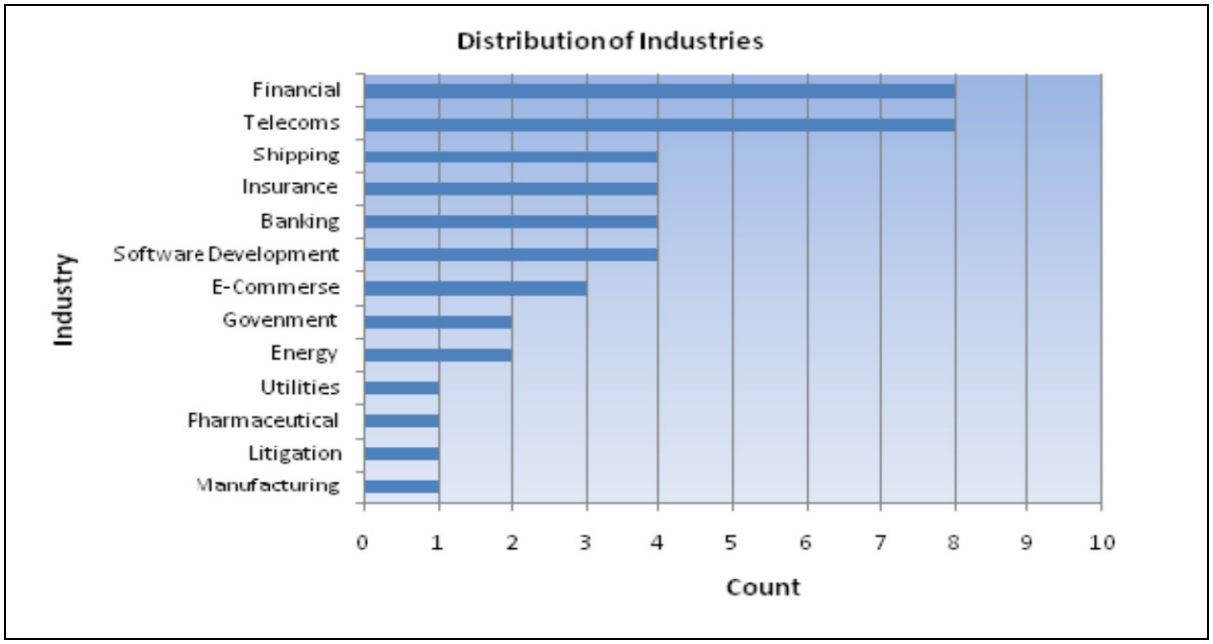

Figure 8: Distribution of industries

Internal Validity: The survey questionnaire was designed with a certain goal in mind; there were no extraneous questions therefore, it is reasonable to conclude that the instrument had internal validity.

3.2 Reliability

Golafshani (2003), asserts "the extent to which results are consistent over time and an accurate representation of the total population under study is referred to as reliability and if the results of a study can be reproduced under a similar methodology, then the research instrument is considered to be reliable." This definition is supported by Miyata and Kai (2009). The survey tool is based upon the Goal-Question Metric Approach template proposed by Basili et al., (1994), the tool demonstrated acceptable levels of reliability in their research therefore, reliability is assumed.

4. Research results

According to the Goal-Question Metric approach (Basili et al.1994) for measurement to be useful from an organisational view point it must be goal driven and specific. The purpose of measurement in this research is to identify the testing tools most likely to overcome organisational constraints and yield optimal test effectiveness.

From the outset two goals were set for this research:

- Identify individual Test Tools (TT) and Test Selection Criteria (TSC) most likely to yield best Test Effectiveness (TE). It was determined that the best statistical approach for identifying these constructs would be correlation analysis.
- Identify the combination of Test Tools (TT) and Test Selection Criteria (TSC) most likely to yield best Test Effectiveness (TE). It was determined that the best statistical approach for identifying these constructs would be multiple regression analysis.

4.1 Descriptives

Constructs for this research were organised into three construct groups:

Software Test Tools (TT) – Table 2.0 lists each tool as presented to respondents, a definition was also added to ensure a common understanding. Tools are listed here with group descriptions where appropriate. Respondents were asked to rate for each project whether the test tool increased test effectiveness. Their responses were graded on a Likert scale of 1 to 5 denoting 'complete disagreement' to 'complete agreement', respectively. Sample size (N=43) in every case.

Table 2: Software test tools

Type Description	Variable Label	Mean	Std. Dev.
Test Design			
Test case content	TT1	3.77	1.269
Test case granularity	TT2	3.33	1.375
Test oracles	TT3	2.63	1.865
Test selection method			
Regression test selection	TT4	3.35	1.526
Re-test all	TT5	2.65	1.462
Test suite reduction	TT6	2.19	1.577
Test case prioritisation	TT7	3.91	1.151
Test execution method			
Automated testing	TT8	2.7	1.859
Manual testing	TT9	4.19	1.22
Smoke testing	TT10	3.6	1.788

Software Test Selection Criteria (TSC) – Table 3.0 list each test selection criteria. Respondents received them with definitions and were asked to rate projects on each criterion; whether these test selection criteria increase software test effectiveness. Responses were graded on a Likert scale of 1 to 5 denoting 'no increase' to 'significantly increased', respectively.

Table 3: Software test selection criteria

Type Description	Variable Label	Mean	Std. Dev.
Test Case			
Code coverage	TSC1	1.4	1.4
Functional coverage	TSC2	3.81	1.118
Defect detection capability	TSC3	3.09	1.702
Setup time	TSC4	2.74	1.706
Implementation complexity	TSC5	2.86	1.207
Execution time	TSC6	3.58	1.239
Requirements			
Customer priority	TSC7	3.63	1.543
Implementation complexity	TSC8	2.88	1.562
Fault proneness	TSC9	2.51	1.907
Volatility	TSC10	2.79	1.767
Test Case Composition			
Test case content	TSC11	3.53	1.182
Test case granularity	TSC12	2.88	1.219
No test selection criteria	TSC13	1.74	1.853

Test Constraints (TC) – Table 4 list the main test constraints encountered by test projects, again respondents were asked to rate projects on each criterion, whether these test constraints hindered software test effectiveness. Responses were graded on a Likert scale of 1 to 5 denoting 'no impact' to 'significantly impacted', respectively.

Table 4: Test constraints

Type Description	Variable Label	Mean	Std. Dev.
Test Constraints			
Time	Time	3.72	1.26
Cost	Costs	2.93	1.534
Skills	Skills	2.81	1.332

The relationship between these variables and test effectiveness can be represented in the following linear function:

Test effectiveness (TE) = Test Tools (TT) + Test Selection Criteria (TSC) – Test Constraints (TC)
Subsequent correlation and regression tests explored the applicability of this function.

4.2 Goal 1 - correlation tests

Pearson correlation was run for each of the constructs independently (assuming two of the three constructs were zero) to see which of the underlying variables correlates with the construct Test Effectiveness (TE).

The results from Table 5 suggest that regression testing selection (TT4) and smoke testing (TT10) correlate strongly with test effectiveness (TE), with 54% and 44% of the variability in TE explained by these two variables, with both achieving significance levels in the 99 percentile, suggesting these results are very reliable. Two further variables, test case prioritisation (TT7) and automated testing (TT8) correlate less strongly with TE (35% and 38%, respectively), but still achieve significance levels in the 95 percentile.

Table 5: Test tools (TT) correlated with TE

Type Description	Variable Label	Pearson Correlation	Significance (2-tailed)
Test Design			
Test case content	TT1	-0.036	0.821
Test case granularity	TT2	0.099	0.527
Test oracles	TT3	0.206	0.185
Test selection method			
Regression test selection	TT4	0.536	0.000**
Re-test all	TT5	0.016	0.919
Test suite reduction	TT6	0.106	0.501
Test case prioritisation	TT7	0.347	0.023*
Test execution method			
Automated testing	TT8	0.384	0.011*
Manual testing	TT9	-0.129	0.409
Smoke testing	TT10	0.444	0.003**

From Table 6 we note that one variable, defect detection capability (TSC3), achieved correlation of 39% at significance levels in the 99 percentile (0.010). Other variables that achieved high significance levels in the 95 percentile are functional coverage, customer priority, test case content and no test selection criteria (test all) with respective correlations of 31%, 35%, 31% and 34%.

Table 6: Test selection criteria (TSC) correlated with TE

Type Description	Variable Label	Pearson Correlation	Significance (2-tailed)
Test Case			
Code coverage	TSC1	0.219	0.159
Functional coverage	TSC2	0.315	0.039*
Defect detection capability	TSC3	0.39	0.010**
Setup time	TSC4	0.27	0.08
Implementation complexity	TSC5	0.196	0.207
Execution time	TSC6	0.071	0.649
Requirements			
Customer priority	TSC7	0.354	0.020*
Implementation complexity	TSC8	0.017	0.914
Fault proneness	TSC9	0.155	0.32
Volatility	TSC10	0.106	0.499
Test Case Composition			
Test case content	TSC11	0.306	0.046*
Test case granularity	TSC12	0.167	0.235
No test selection criteria	TSC13	0.343	0.024*
Table 6: Test Selection Criteria (TSC) correlated with TE			

There were no significant correlations with any variables in the test constraint construct (TC).

4.3 Goal 2 – multiple regression testing

Having identified variables to emerge from independent tests of association with Test Effectiveness, attention was turned to testing the linear function in totality using multiple regression analysis to see if the variables that emerged in the correlation test would hold or would new variables emerge as predictors of test effectiveness.

Data for the model were analysed using linear multiple regression analysis. The procedure estimates the coefficients (*beta*) of one or more independent variables to predict (R^2) the value of a single dependent variable. Variables are systematically entered and removed from the equation using the stepwise method to determine the line of best fit.

The model depicted in table 7 suggests that 48% of the variability in the construct Test Effectiveness (TE) is explained by the variables TT4, TT10 and TSC13. The proportion that each of the independent variables explain is indicated by their beta percentages, 46%, 34% & 33% respectively.

Table 7: Linear regression model

Dependent Variable	Independent Variables	Beta	R	Adjusted R^2	Significance
Test Effectiveness (TE)			0.718	0.479	0
	Regression test selection (TT4)	0.465			0
	Smoke testing (TT10)	0.34			0.005
	No test selection criteria (TSC13)	0.337			0.004

5. Discussion

Test tools and test selection are applied quite extensively in the test cases. Given the diversity of the population it can be assumed this is likely to be the case for the Software Testing industry globally. This indicates that software testing practice is becoming more rigorous and formalised and to a certain extent scientific, which is a good sign for the software development industry and business as a whole.

The results suggest regression test selection (TT4) and smoke testing (TT10) significantly, and to a lesser extent test case prioritisation (TT7) and automated testing (TT11) correlated with Test Effectiveness. These Test Tools can each be applied individually to testing activities in order to increase Test Effectiveness.

In practice these Test Tools are most associated with increased Test Effectiveness and their use is more likely to increase the probability that a software development project will meet its strategic goals.

Smoke testing is used to detect defects before more expensive formal testing. Automated testing significantly reduces test execution time and also increases the rate of functional coverage during testing thus reducing test cycle time. Individually the application of these Test Tools will lead to early detection of defects, reduced cost of testing, quicker product time to market and increased software quality.

Regression test selection and test case prioritisation are dependent upon adequate test selection criteria. One test selection criterion that correlated strongly with Test Effectiveness is defect detection capability (TSC3). Other test selection criteria that correlated with slightly less significance are functional coverage (TSC2), customer priority (TSC7), test case content (TSC11) and no test selection criteria (TSC13). Defect detection capability (TSC3) is an imperative design consideration. It is the make or break factor in the realisation of software quality through increased test effectiveness. It plays a pivotal role in reducing the risk of software going to market with undetected defects. Maximised functional coverage (TSC2) during test execution reduces the risk of parts of a software system being untested after it has been released to market. Test case content (TSC11) directly relates to functional coverage and as such is used to measure functional coverage of software testing, thus giving the business a risk based view that can be used to determine the readiness of a software system before release to market. Customer priority (TSC7) allows software development businesses to quickly meet the most important goals of the customer. No test selection criteria (TSC13) translates into testing of the entire software system thereby mitigating all risk posed by inadequate functional coverage. However this is an expensive approach to software testing and requires no or very little time constraints.

Individually these Test Selection Criteria increase software quality, customer satisfaction thereby possibly enhancing and preserving a business's good reputation.

Applying multiple regression analysis to the combined Test Tools (TT) and Test Selection Criteria (TSC) identified the combined variables of regression test selection (TT4), smoke testing (TT10) and no test selection criteria (TSC13) to have the most significant impact upon Test Effectiveness.

The data analysis strongly suggested a *saturation point* in the application of the number of Test Tools and Test Selection Criteria, exactly at which point this achieved is debatable but initial indications from this research suggests after a 80% level of Test Effectiveness has been achieved. The total number of Test Tools and Test Selection Criteria in this study is twenty-three (23). Correlation and multiple regression analyses reduced this to nine and three respectively, suggesting a high-degree of over-lap between tests. In business terms it would seem not to make sense to try and achieve levels of Test Effectiveness beyond the 80 percentile point.

Interestingly, neither correlation nor multiple regression analyses revealed a significant relationship between Test Constraints and Test Effectiveness. Given that organisational constraints are always uppermost in practitioner's minds. One reason for this absence could be due to the fact that the sample was made up of seasoned test experts, who through their experience have learnt how to mitigate the impact of Test Constraints on Test Effectiveness.

6. Conclusion

The aim of this paper was to identify from the plethora of Test Tools used in practice:

- Which *individual* test tools are most likely to yield optimal test effectiveness and,
- Which *combination* of test tools is most likely to yield optimal test effectiveness and mitigate the effect of test constraints

Data from forty three cases across various industries and countries were collected to identify the current application of Test Tools in practise. The research was designed using the Goal-Question

Metric approach and the data analysed using correlation analysis to identify *individual* Test Tools and Test Selection Criteria most closely associated with Test Effectiveness and; multiple regression analysis to identify the combination of Test Tools and Test Selection Criteria that would lead to optimum Test Effectiveness.

The correlation analysis identified nine variables; Four Test Tools (TT): regression test selection (TT4), smoke testing (TT10), test case prioritisation (TT7) and automated testing (TT11) and, five Test Selection Criteria (TSC): defect detection capability (TSC3), functional coverage (TSC2), customer priority (TSC7), test case content (TSC11) and no test selection criteria (TSC13), allowing test practitioners to pick and mix the various approaches given the specific constraints. The multiple regression analysis identified the combined variables of regression test selection (TT4), smoke testing (TT10) and no test selection criteria (TSC13) to have the most significant impact upon Test Effectiveness.

For practitioners the main value of this research is that it begins to spell out which individual and combined test tools will most likely assist in achieving optimal test effectiveness in the presence of test constraints. Based on the study results approximately 50% of the test effectiveness results are achieved through a combination of regression test selection, smoke testing and no test selection criteria.

Individually regression test selection and smoke testing correlated reliably with test effectiveness and less significantly test case prioritisation and automated testing. Of the test selection criteria: defect detection capability, functional coverage, customer priority, test case content and no test selection criteria (test all) correlated reliably with test effectiveness.

In practice this suggest that smoke testing should be used for early detection of major defects after which regression test selection should be applied to the software under test and that no test selection be applied to test cases, therefore test everything. This only makes sense when there are no time constraints. Smoke testing is commonly used to determine if a system is ready for formal testing. This assists with effective resource allocation. If smoke testing is part of a build process that is run daily or nightly the project benefits from early detection of defects. This reduces the cost of defect resolution.

Testing everything is an approach best used for major software releases and mission critical applications. Though this is expensive it mitigates most of the risk of defects 'making it out into the wild' which could result in damaged reputation and possible significant economic loss and even loss of life. During minor releases or when time, cost or both are constraints it does not make economic sense to do exhaustive testing with full functional coverage. In this case one or more of the test selection methods supported by appropriate test selection criteria identified through the correlation analysis should be applied. This will lead to reduced test execution costs and time with increased defect detection rate in system under test. Regression test selection aims only to run tests where changes have been made therefore as test selection criteria functional coverage and test case content makes sense. Test case prioritisation is used to achieve specific goals such as find as many defect as possible or test critical customer components in these cases defect detection capability and customer priority as test selection criteria makes sense. Automated testing is a good option to decrease test execution time. However it has become frequent in literature to read about failed automated testing initiatives**Error! Reference source not found.** ; Ramler and Wolfmaier 2006). This has not been discussed in this paper however practitioners will do well to do thorough investigation in automated testing before embarking upon a test automation endeavour.

Areas for further research:

- A saturation point in the application of Test Tools and Test selection criteria is alluded to in this paper, more evidence to support this idea is required and to pinpoint the threshold. This will enable test practitioners achieve greater precision when trading-off the cost of further software testing with relative benefits.
- Of the research sample the majority of test cases were overwhelmingly successful, further research needs to be conducted on cases which are not successful to ascertain if the emergent variables remain consistent.

References

Baresi, L., Pezze`, M. 2006, 'An Introduction to Software Testing', *Electronic Notes in Theoretical Computer Science*, 2006.

Basili, V.R. Caldiera, G., Rombach, H.D. 1994, 'The Goal Question Metric Approach'.

Berner, S., Weber, R., Keller, R.K. 2005, 'Observations and Lessons Learned from Automated Testing', *Zühlke Engineering AG Zürich-Schlieren Switzerland*, 2005.

Bertolino, A. 2007, 'Software Testing Research: Achievements, Challenges, Dreams', *Future of Software Engineering (FOSE'07)*, 2007.

Bryce, R.C., Colbourn, C.J. 2005, 'Test Prioritization for Pairwise Interaction Coverage', *A-MOST'05*, Copyright 2005 ACM 1-59593-115-5/00/0004, St. Louis, Missouri, USA.

Do, H., Mirarab, S., Tahvildari, L. 2008, 'An Empirical Study of the Effect of Time Constraints on the Cost-Benefits of Regression Testing', *SIGSOFT 2008/FSE-16, November 9–15*, Copyright 2008 ACM 978-1-59593-995-1, Atlanta, Georgia, USA.

Engel, A., Last, M. 2007, 'Modeling software testing costs and risks using fuzzy logic paradigm', *The Journal of Systems and Software*, 2007.

Engström, E 2010, 'Regression Test Selection and Product Line System Testing', *2010 Third International Conference on Software Testing, Verification and Validation*, IEEE Computer Society, Paris.

Golafshani, N 2003, 'Understanding Reliability and Validity in Qualitative Research', *The Qualitative Report*, vol 8, no. 4, pp. 597-607.

Harman, M. 2008, 'Open Problems in Testability Transformation', *IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08)*, IEEE Computer Society.

Huang, C-Y 2005, 'Performance analysis of software reliability growth models with testing-effort and change-point', *The Journal of Systems and Software*, vol 76, pp. 181 - 194.

Karhu, K., Repo, T., Taipale, O., Smolander, K. 2009, 'Empirical Observations on Software Testing Automation'.

McMaster, S., Memon, A.M. 2008, 'Call-Stack Coverage for GUI Test Suite Reduction', *IEEE Transactions on Software Engineering*, 2008.

McMinn, P 2009, 'Search-Based Failure Discovery using Testability Transformations to Generate Pseudo-Oracles', *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ACM, Montreal Quebec, Canada.

Memon, A.M., Bananerjee, I., Hashmi, N., Nagarajan, A. 2003, 'DART: A Framework for Regression Testing Nightly/daily Builds of GUI Applications', *Proceedings of the International Conference on Software Maintenance (ICSM'03)*, IEEE Computer Society.

Memon, A.M., Xie, Q. 2004, 'Empirical evaluation of the fault-detection effectiveness of smoke regression test cases for GUI-based software', *Proceedings of the 20th International Conference on software Maintenance (ICSM 2004)*, Chicago, USA.

Memon, A.M., Xie., Q. 2005, 'Studying the Fault-Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software', *IEEE Transactions on Software Engineering*, 2005.

Miyata, H & Kai, I 2009, 'Reconsidering Evaluation Criteria for Scientific Adequacy in Health Care Research : An Integrative Framework of Quantitative and Qualitative Criteria', *International Journal of Qualitative Methods*, vol 8, no. 1, pp. 64-75.

Parse, S., Khalilian, A., Fazlalizadeh, Y. 2009, 'A New Algorithm to Test Suite Reduction Based on Cluster Analysis', 2009.

Persson, C & Yilmaztürk, N 2004, 'Establishment of automated regression testing at ABB: industrial experience report on 'avoiding the pitfalls'', *Proceedings. 19th International Conference on Automated Software Engineering, 2004.*, IEEE Computer Society, Linz, Austria.

Ramler, R., Wolfmaier, K. 2006, 'Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost'".

Rothermel, G., Elbaum, S., Malishevsky, A.G. , Kallakuri, P., Xuemei Q. 2004, 'On Test Suite Composition and Cost-Effective Regression Testing', *ACM Transactions on Software Engineering and Methodology*, 2004.

Sampath, S., Ren´ee C. B., Gokulanand, V., Vani K., A. Gunes¸ K. 2008, 'Prioritizing User-session-based Test Cases for Web Applications Testing', *International Conference on Software Testing, Verification, and Validation*, IEEE Computer Society.

Shahamiri, S.R., Kadir, W.M.N.W, Mohd-Hashim. S.Z. 2009, 'A Comparative Study on Automated Software Test Oracle Methods', *Fourth International Conference on Software Engineering Advances*, IEEE Computer Society.

Srikanth, H., Williams, L. 2005, 'On the Economics of Requirements-Based Test Case Prioritization', *EDSER'05*, Copyright 2005 ACM 1-59593-118-X/05/0005, St. Louis, Missouri, USA.

Srikanth, H., Williams, L., Osborne, J. 2005, 'System Test Case Prioritization of New and Regression Test Cases', *International Symposium on Empirical Software Engineering*, IEEE Computer Society.

Vallespir, D., Herbert, J. 2009, 'Effectiveness and Cost of Verification Techniques', IEEE Computer Society, Mexican International Conference on Computer Science.

Zhang, R., Jiang, J., Yin, J., Jin, A., Lou, J., Wu, Y. 2008, 'A New Method for Test Suite Reduction', *The 9th International Conference for Young Computer Scientists*, IEEE Computer Society.

Zhang, X., Xu, B., Chen, Z., Nie. C., Li, L. 2008, 'An Empirical Evaluation of Test Suite Reduction for Boolean Specification-based Testing', *The Eighth International Conference on Quality Software*, IEEE Computer Society.

