A Systematic Review on Regression Test Selection Techniques

Emelie Engström
Department of Computer Science
Lund University
SE-221 00 LUND
+46 46 222 88 99
emelie.engstrom@cs.lth.se

Per Runeson
Department of Computer Science
Lund University
SE-221 00 LUND
+46 46 222 93 25
per.runeson@cs.lth.se

Mats Skoglund
Department of Computer Science
Lund University
SE-221 00 LUND
mats.skoglund@cs.lth.se

ABSTRACT

Regression testing is verifying that previously functioning software remains after a change. With the goal of finding a basis for further research in a joint industry-academia research project, we conducted a systematic review of empirical evaluations of regression test selection techniques. We identified 27 papers reporting 36 empirical studies, 21 experiments and 15 case studies. In total 28 techniques for regression test selection are evaluated. We present a qualitative analysis of the findings, an overview of techniques for regression test selection and related empirical evidence. No technique was found clearly superior since the results depend on many varying factors. We identified a need for empirical studies where concepts are evaluated rather than small variations in technical implementations.

1 INTRODUCTION

Efficient regression testing is important, even crucial, for organizations with a large share of their cost in software development. It includes, among other tasks, determining which test cases need to be re-executed, i.e. regression test selection, in order to verify the behavior of modified software. Regression test selection involves a trade-off between the cost for re-executing test cases, and the risk for missing faults introduced through side effects of changes to the software. Iterative development strategies and reuse are common means of saving time and effort for the development. However they both require frequent retesting of previously tested functions due to changes in related code. The need for efficient regression testing strategies is thus becoming more and more important.

A great deal of research effort has been spent on finding cost-efficient methods for different aspects of regression testing. Examples include test case selection based on code changes [1][6][13][17][20][22][43][49][62][64][67] and specification changes [38][40][54][68], evaluation of selection techniques [48], change impact analysis [44], regression tests for different applications e.g. database applications [18], regression testing of GUIs and test automation [39], and test process enhancement[31]. To

bring structure to the topics, researchers have typically divided the field of regression testing into i) test selection, ii) modification identification, iii) test execution, and iv) test suite maintenance. This review is focused on test selection techniques for regression testing.

Although techniques for regression test selection have been evaluated in previous work[3][15][36][65], no general solution has been put forward since no technique could possibly respond adequately to the complexity of the problem and the great diversity in requirements and preconditions in software systems and development organizations. Neither does any single study evaluate every aspect of the problem; e.g. Kim *et al.* [27] evaluate the effects of regression test application frequency, Elbaum *et al.* [11] investigate the impact that different modifications have on regression test selection techniques, several studies examine the ability to reduce regression testing effort [3][11][15][27][36][65][66] and to reveal faults [11][15][27][49].

In order to map the existing knowledge in the field, we launched a systematic review to collect and compare existing empirical evidence on regression test selection. The use of systematic reviews in the software engineering domain has been subject to a growing interest in the last years. In 2004 Kitchenham proposed a guideline adapted to the specific characteristics of software engineering research. This guideline has been followed and evaluated [5][30][57] and updated accordingly in 2007 [29]. Kitchenham *et al.* recently published a review of 20 systematic reviews in software engineering 2004-2007[28].

Ideally, several empirical studies identified in a systematic review evaluate the same set of techniques under similar conditions on different subject programs. Then there would be a possibility to perform an aggregation of findings or even meta-analysis and thus enable drawing general conclusions. However, as the field of empirical software engineering is quite immature, systematic reviews have not given very clear pictures of the results. In this review we found that the existing studies were diverse, thus hindering proper quantitative aggregation. Instead we present a qualitative analysis of the findings, an overview of existing techniques for regression test selection and of the amount and quality of empirical evidence.

There are surveys and reviews of software testing research published before, but none of these has the broad scope and the extensive approach of a systematic review. In 2004 Do *et al*. presented a survey of empirical studies in software testing in general [8] including regression testing. Their study covered two journals and four conferences over ten years (1994-2003). Other reviews of regression test selection are not exhaustive but compare a limited number of chosen regression test selection techniques. Rothermel and Harrold presented a framework for evaluating regression test techniques already in 1996 [48] and evaluated the, by that time, existing techniques. Juristo *et al.* aggregated results from unit testing experiments [25] of which some evaluate regression testing techniques, although with a more narrow scope. Binkley *et al.* reviewed research on the

application of program slicing to the problem of regression testing [4]. Hartman *et al.* reports a survey and critical assessment of regression testing tools [21]. However, as far as we know, no systematic review on regression test selection research has been carried through since the one in 1996 [48]. An early report of this study was published in 2008 [12], which here is further advanced especially with respect to the detailed description of the techniques (Section 3.4), their development history and the analysis of the primary studies (Section 3.5).[1]

This paper is organized as follows. In section 2 the research method used for our study is described. Section 3 reports the empirical studies and our analyses. Section 4 discusses the results and section 5 concludes the work.

2 RESEARCH METHOD

2.1 Research Questions

This systematic review aims at summarizing the current state of the art in regression test selection research by proposing answers to a set of questions below. The research questions stem from a joint industry-academia research project, which aims at finding efficient procedures for regression testing in practice. We searched for candidate regression test selection techniques that were empirically evaluated, and in case of lack of such techniques, to identify needs for future research. Further, as the focus is on industrial use, issues of scale-up to real-size projects and products are important in our review. The questions are:

RQ1) Which techniques for regression test selection in the literature have been evaluated empirically?

RQ2) Can these techniques be classified, and if so, how?

RQ3) Are there significant differences between these techniques that can be established using empirical evidence?

RQ4) Can technique *A* be shown to be superior to technique *B*, based on empirical evidence?

Answers to these research questions are searched in the published literature using the procedures of systematic literature reviews as proposed by Kitchenham [29].

[1] In this extended analysis, some techniques that originally were considered different ones, were considered the same technique. Hence, the number of techniques differ from [10]. Further, the quality of two empirical studies was found insufficient in the advanced analysis, why two studies were removed.

2.2 Sources of information

In order to gain a broad perspective, as recommended in Kitchenham's guidelines [29], we searched widely in electronic sources. The advantage of searching databases rather than a limited set of journals and conference proceedings, is also empirically motivated by Dieste *et al.* [7]. The following seven databases were covered:

- Inspec (<www.theiet.org/publishing/inspec>)
- Compendex (<www.engineeringvillage2.org>)
- ACM Digital Library (<portal.acm.org>)
- IEEE eXplore (<ieeexplore.ieee.org>)
- ScienceDirect (<www.sciencedirect.com>)
- Springer LNCS (<www.springer.com/lncs>)
- Web of Science(<www.isiknowledge.com>)

These databases cover the most relevant journals and conference and workshop proceedings within software engineering, as confirmed by Dybå *et al.* [8]. Grey literature (technical reports, some workshop reports, work in progress) was excluded from the analysis for two reasons: the quality of the grey literature is more difficult to assess and the volume of studies included in the first searches would have grown unreasonably. The searches in the sources selected resulted in overlap among the papers, where the duplicates were excluded primarily by manual filtering.

2.3 Search criteria

The initial search criteria were broad in order to include articles with different uses of terminology. The key words used were <regression> and (<test> or <testing>) and <software>, and the database fields of title and abstract were searched. The start year was set to 1969 to ensure that most relevant research within the field would be included, and the last date for inclusion is publications within 2006. The earliest primary study actually included was published in 1997. Kitchenham recommends that exclusion based on languages should be avoided [29]. However, only papers written in English are included. The initial search located 2 923 potentially relevant papers.

2.4 Study Selection

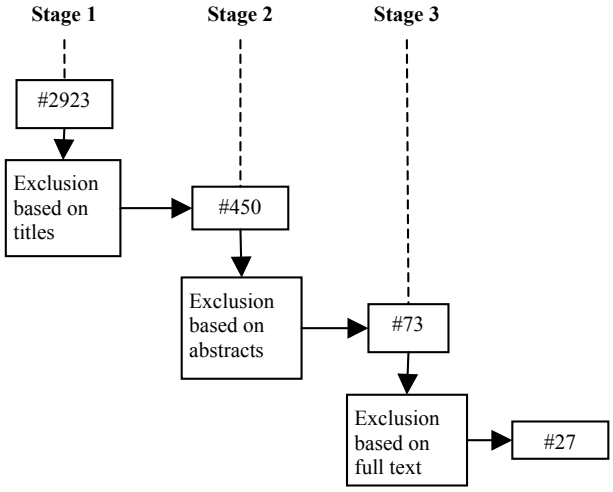


Figure 1. Study selection procedure

In order to obtain independent assessments, four researchers were involved in a three-stage selection process, as depicted in Figure 1.

In the first stage duplicates and irrelevant papers were excluded manually based on titles. In our case, the share of irrelevant papers was extremely large since papers on software for *statistical* regression testing or other regression testing could not be distinguished from papers on *software* regression testing in the database search. The term software did not distinguish between the two areas, since researchers on statistical regression testing often develop some software for their regression test procedures. After the first stage 450 papers remained.

In the second stage, information in abstracts was analyzed and the papers were classified along two dimensions: research approach and regression testing approach. Research approaches were experiment, case study, survey, review, theory and simulation. The two latter types were excluded, as they are not presenting an empirical research approach, and the survey and review papers were not considered as being primary studies but rather related work to the systematic review. At this stage we did not judge the quality of the empirical data. Regression testing approaches were selection, reduction, prioritization, generation, execution and other. Only papers focusing on regression test *selection* were included.

In the third stage a full text analysis was performed on the 73 papers and the empirical quality of the studies was further assessed. The following questions were asked in order to form quality criteria for which studies to exclude before the final data extraction:

- Is the study focused on a specific regression test selection method? E.g. a paper could be excluded that presents a method that potentially could be used for regression testing, but is evaluated from another point of view..
- Are the metrics used and the results relevant for a comparison of methods? E.g. a paper could be excluded which only reports on the ability to predict fault prone parts of the code, but not on the fault detection effectiveness or the cost of the regression test selection strategy.
- Is data collected and analyzed in a sufficiently rigorous manner? E.g. a paper could be excluded if a subset of components was analyzed and conclusions were drawn based on those, without any motivation for the selection.

These questions are derived from a list of questions, used for a similar purpose, published by Dybå *et al.* [8]. However in our review context, quality requirements for inclusion had to be weaker than suggested by Dybå *et al.* in order to obtain a useful set of studies to compare. The selection strategy was in general more inclusive than exclusive. Only papers with very poorly reported or poorly conducted studies were excluded, as well as papers where the comparisons made were considered irrelevant to the original goals of this study.

Abstract analysis and full text analysis were performed in a slightly iterative fashion. Firstly, the articles were independently assessed by two of the researchers. In case of disagreement, the third researcher acted as a checker. In many cases, disagreement was due to insufficient specification of the criteria. Hence, the criteria were refined and the analysis was continued.

In order to get a measure of agreement in the study selection procedure, the Kappa coefficient was calculated for the second stage, which comprised most judgments in the selection. In the second stage 450 abstracts were assessed by two researchers independently. In 41 cases conflicting assessments were made which corresponds to the Kappa coefficient K = 0,78. According to Landis and Koch [33] this translates to a substantial strength of agreement.

2.5 Data extraction and synthesis

Using the procedure, described in the previous section, 27 articles were finally selected that reported on 36 unique empirical studies, evaluating 28 different techniques. The definition of what constitutes a single empirical study, and what constitutes a unique technique is not always clear cut. The following definitions have been used in our study:

- Study: an empirical study applying a technique to one or more programs. Decisions on whether to split studies with multiple artifacts into different studies were based on the authors' own classification

of the primary studies. Mostly, papers including studies on both small and large programs are presented as two different studies.

- Technique: An empirically evaluated method for regression test selection. If the only difference between two methods is an adaption to a specific programming language (e.g. from C++ to Java) they are considered being the same technique.

Studies were classified according to type and size, see Section 3.1. Two types of studies are included in our review, experiments and case studies. We use the following definitions:

- Experiment: A study in which an intervention is deliberately introduced to observe its effects [55].

- Case study: An empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between the phenomenon and context are not clearly evident [69].

Surveys and literature reviews were also considered in the systematic review, e.g. [48] and [25], but rather as reference point for inclusion of primary studies than as primary studies as such.

Regarding size, the studies are classified as small, medium or large (S, M, L) depending on the study artifact sizes. A small study artifact has less than 2,000 lines of code (LOC), a large study artifact has more than 100,000 LOC, and a medium sized study artifact is in between. The class limits are somewhat arbitrarily defined. In most of the articles the lines of code metric is clearly reported and thus this is our main measurement of size. But in some articles sizes are reported in terms of number of methods or modules, reported as the authors' own statement about the size or not reported at all.

The classification of the techniques is part of answering RQ2 and is further elaborated in Section 3.4.

2.6 Qualitative assessment of empirical results

The results from the different studies were qualitatively analyzed in categories of four key metrics: reduction of cost for test execution, cost for test case selection, total cost, and fault detection effectiveness, see Section 3.5.2. The "weight" of an empirical study was classified according to the scheme in Table 1. A study with more "weight" is considered contributing more to the overall conclusions. A unit of analysis in an experiment is mostly a version of a piece of code, while in a case study; it is mostly a version of a whole system or sub-system.

Table 1. "Weight" of empirical study.

Type and size of study	Light empirical study "weight"	Medium empirical study "weight"
Experiment (small) Case study (small-medium)	Analysis units < 10	Analysis units >= 10
Experiment (medium) Case study (large)	Analysis units < 4	Analysis units >= 4

The results from the different studies were then divided into six different categories according to the classification scheme in Table 2. The classification is based on the study "weight" and the size of the difference in a comparative empirical study. As the effect sizes were rarely reported in the studies, the sizes of the differences are also qualitatively assessed. The categorization of results was made by two researchers in parallel and uncertainties were resolved in discussions. Results are presented in Figures 5 – 8 in Section 3.5.

Table 2. Classification scheme for qualitative assessment of the weight of empirical results.

	No difference	Difference of small size	Difference of large size
Medium empirical study "weight"	Strong indication of equivalence between the two compared techniques	Weak indication that one technique is superior to the other	Strong indication that one technique is superior to the other
Light empirical study "weight"	Weak indication of equivalence between the two compared techniques	No indication of differences or similarities	Weak indication that one technique is superior to the other

2.7 Threats to validity

Threats to the validity of the systematic review are analyzed according to the following taxonomy; construct validity, reliability, internal validity and external validity.

Construct validity reflects to what extent the phenomenon under study really represents what the researchers have in mind and what is investigated according to the research questions. The main threat here is related to terminology. Since the systematic review is based on a hierarchical structure of terms – regression test/testing consists of the activities modification identification, test selection, test execution and test suite maintenance – we might miss other relevant studies on test selection that are not specifically aimed for regression testing. However, this is a consciously decided limitation, which has to be taken into account in the use of the results. Another aspect of the construct validity is assurance that we actually find all papers on the selected topic. We

analyzed the list of publication fora and the list of authors of the primary studies to validate that no major forum or author was missed.

Reliability focuses on whether the data is collected and the analysis is conducted in a way that it can be repeated by other researchers with the same results. We defined a study protocol setting up the overall research questions, the overall structure of the study as well as initial definitions of criteria for inclusions/exclusion, classification and quality. The criteria were refined during the study based on the identification of ambiguity that could mislead the researchers.

In a systematic review, the decision process for inclusion and exclusion of primary studies is the major focus when it comes to reliability, especially in this case where another domain (statistics) also uses the term regression testing. Our countermeasures taken to reduce the reliability threat were to set up criteria and to use two researchers to classify papers in stages 2 and 3. In cases of disagreement, a third opinion is used. However, the Kappa analysis indicates strong agreements. One of the primary researchers was changed between stages 2 and 3. Still, the uncertainties in the classifications are prevalent and a major threat to reliability, especially since the quality standards for empirical studies in software engineering are not high enough. Research databases is another threat to reliability [8]. The threat is reduced by using multiple databases; still the non-determinism of some database searches is a major threat to the reliability of any systematic review.

Internal validity is concerned with the analysis of the data. Since no statistical analysis was possible due to the inconsistencies between studies, the analysis is mostly qualitative. Hence we link the conclusions as clearly as possible to the studies, which underpin our discussions.

External validity is about generalizations of the findings derived from the primary studies. Most studies are conducted on small programs and hence generalizing them to a full industry context is not possible. In the few cases were experiments are conducted in the small as well as case studies in the large, the external validity is reasonable, although there is room for substantial improvements.

3 RESULTS

3.1 Primary studies

The goal of this study was to find regression test selection techniques that are empirically evaluated. The papers were initially obtained in a broad search in seven databases covering relevant journals, conference and workshop proceedings within software engineering. Then an extensive systematic selection process was carried out to identify papers describing empirical evaluations of regression test selection techniques. The results presented here thus give a good picture of the existing evidence base.

Out of 2 923 titles initially screened, 27 papers (P1-P27) on empirical evaluations of techniques for regression test selection remained until the final stage. These 27 papers report on 36 unique studies (S1-S36), see Table 3, and compare in total 28 different techniques for regression test selection for evaluation (T1-T28), see listing in Table 8 below, which constitutes the primary studies of this systematic review. Five reference techniques are also identified (REF1-REF5), e.g. *re-test all* (all test cases are selected) and *random(25)* (25% of the test cases are randomly selected). In case the studies are reported partially or fully in different papers, we generally refer to the most recent one as this contains the most updated study. When referring to the techniques, we do on the contrary refer to the oldest, considering it being the original presentation of the technique.

Table 3. Primary studies, S1-S36, published in papers P1-P27, evaluation techniques T1-T28.

Study ID	Publica-tion ID	Reference	Techniques	Artifacts	Type of study	Size of study
S1	P1	Baradhi and Mansour (1997) [2]	T4, T5, T6, T11, T12 REF1	Own unspecified	Exp	S
S2	P2	Bible et al. (2001) [3]	T7, T8 REF1	*7x Siemens,* Small constructed programs, constructed, realistic non-coverage based test suites	Exp	S
S3	P2	Bible et al. (2001)[3]	T7, T8 REF1	*Space,* Real application, real faults, constructed test cases	Exp	S
S4	P2	Bible et al. (2001) [3]	T7, T8 REF1	*Player*, One module of a large software system constructed realistic test suites	Exp	M
S5	P3	Elbaum et al. (2003) [11]	T2, T4, T18 REF1	*Bash, Grep, Flex* and *Gzip*, Real, non-trivial C program, constructed test suites	CS (Mult)	M
S6	P4	Frankl et al. (2003) [14]	T7, T10 REF1	*7xSiemens,* Small constructed programs, constructed, realistic, non-coverage based test suites	Exp	S
S7	P5	Graves et al. (2001) [15]	T1, T2, T7 REF1, REF2, REF3, REF4	*7xSiemens*, Small constructed programs, constructed, realistic non-coverage based test suites; *space,* Real application, real faults, constructed test cases*; player*, One module of a large software system constructed realistic test suites	Exp	S M
S8	P6	Harrold et al. (2001) [19]	T15 REF1	*Siena, Jedit, JMeter, RegExp,* Real programs, constructed faults	Exp	S
S9	P7	Kim et al. (2005)[27]	T2, T7, T8 REF1, REF2, REF3, REF4	*7xSiemens*, Small constructed programs, constructed, realistic non-coverage based test suites; *Space,* Real application, real faults, constructed test cases	Exp	S
S10	P8	Koju et al. (2003) [32]	T15 REF1	*Classes in .net framework,* Open source, real test cases	Exp	S
S11	P9	Mansour et al. (2001) [36]	T4, T5, T6, T12	20 small sized Modules	Exp	S
S12	P10	Mao and Lu (2005) [38]	T16, T17, T24 REF1	*Triangle, eBookShop, ShipDemo,* Small Constructed programs	CS	S
S13	P11	Orso et al. (2004) [41]	T9, T15, T19 REF1	*Jaba, Daikon, JBoss,* Real-life programs, original test suites	Exp	M L

S14	P12	Pasala and Bhowmick (2005) [42]	T20 REF1	*Internet Explorer* (client), *IIS* (web server), *application* (app. Server), An existing browser based system, real test cases	CS	NR
S15	P13	Rothermel and Harrold (1997) [49]	T7 REF1	*7xSiemens*, Small constructed programs, constructed, realistic non-coverage based test suites	Exp	S
S16	P13	Rothermel and Harrold (1997) [49]	T7 REF1	*Player*, One module of a large software system constructed realistic test suites	Exp	M
S17	P14	Rothermel and Harrold (1998) [50]	T7 REF1	*7xSiemens,* Small constructed programs, constructed, realistic non-coverage based test suites	Exp	S
S18	P14	Rothermel and Harrold (1998) [50]	T7 REF1	*7xSiemens,* Small constructed programs, constructed, realistic non-coverage based test suites	Exp	S
S19	P14	Rothermel and Harrold (1998) [50]	T7 REF1	*7xSiemens*, Small constructed programs, constructed, realistic non-coverage based test suites;	Exp	S
S20	P14	Rothermel and Harrold (1998) [50]	T7 REF1	*Player,* One module of a large software system constructed realistic test suites	Exp	M
S21	P14	Rothermel and Harrold (1998) [50]	T7 REF1	*Commerercial,* Real application, real test suite	Exp	S
S22	P15	Rothermel et al. (2002)[45]	T8, T18 REF1	*Emp-server,* Open-source, server component, constructed test cases; *Bash* Open-source, real and constructed test cases	Exp	M
S23	P16	Rothermel et al. (2004)[46]	T2, T8, T18 REF1	*Bash,* Open-source, real and constructed test cases	Exp	M
S24	P16	Rothermel et al. (2004) [46]	T2, T8, T18 REF1	*Emp-server,* Open-source, server component, constructed test cases	Exp	M
S25	P17	Skoglund and Runeson (2005) [56]	T9, T21 REF1	*Swedbank,* Real, large scale, distributed, component-based, J2EE system, constructed, scenario-based test cases	CS	L
S26	P18	Vokolos and Frankl (1998) [65]	T10 REF1	*ORACOLO2,* Real industrial subsystems, real modifications, constructed test cases	CS	M
S27	P19	White and Robinson (2004) [61]	T3 REF5	*14 real ABB projects,* Industrial, Real-time system	CS	L
S28	P19	White and Robinson (2004) [61]	T9 REF5	*2 real ABB projects*, Industrial, Real-time system	CS	L
S29	P20	White et al. (2005) [60]	T3, T9, T25	OO-telecommunication software system	CS	S
S30	P20	White et al. (2005) [60]	T3, T9, T25	OO – real-time software system	CS	L
S31	P21	Willmor and Embury (2005)[63]	T7, T22, T23 REF1	*Compiere, James, Mp3cd browser,* Open source systems, real modifications	CS	NR
S32	P22	Wong et al. (1997)[66]	T13 REF1	*Space,* Real application, real faults, constructed test cases	CS	S
S33	P23	Wu et al. (1999) [67]	T14 REF1	*ATM-simulator,* small constructed program	CS	S
S34	P23	Wu et al. (1999) [67]	T14 REF1	Subsystem of a fully networked supervisory control and data analysis system	CS	M
S35	P24, P25, P26	Zheng et al. (2005) [71], Zheng et al. (2006) [72] Zheng (2005) [70]	T26, T28 REF1	*ABB-internal,* Real C/C++ application	CS	M
S36	P27, P25	Zheng et al. (2006) [74], Zheng et al. (2006) [72]	T27, T28 REF1	*ABB-internal,* Real C/C++ application	CS	M

In most of the studies, the analyses are based on descriptive statistics. Tabulated data or bar charts are used as a basis for the conclusions. In two studies (S23 and S24), published in the same paper (P16) [46] statistical analysis is conducted, using ANOVA.

3.2 Analyses of the primary studies

In order to explore the progress of the research field, and to validate that the selected primary studies reasonably cover our general expectations of which fora and which authors should be represented, we analyze, as an extension to RQ1, aspects of the primary studies as such: where they are published, who published them, and when. As defined in Section 2.5, a paper may report on multiple studies, and in some cases the same study is reported in more than one paper. Different researchers have different criteria for what constitutes a study. We have tried to apply a consistent definition of what constitutes a study. This distribution of studies over papers is shown in Table 4. Most papers (18 out of 27) report a single study, while few papers report more than one. Two papers report new analyses of earlier published studies. Note that many of the techniques are originally presented in papers without empirical evaluation, hence these papers are not included as primary studies in the systematic review, but referenced in Section 3.3 as sources of information about the techniques as such (Table 8).

Table 4. Distribution of number of papers after the number of studies each paper reports

# reported studies in each paper	# papers	# studies
0 (re-analysis of another study)	2	0
1	18	18
2	5	10
3	1	3
5	1	5
Total	**27**	**36**

The number of identified techniques in the primary studies is relatively high compared to the number of studies, 28 techniques were evaluated in 36 studies. Table 5 presents the distribution of number of studies in which different techniques occur. One technique was present in 14 different studies, another technique in 8 studies etc. 14 techniques only appear in one study, which is not satisfactory when trying to aggregate information from empirical evaluations of the techniques.

Table 5. Distribution of techniques after occurrences in number of studies

Represented in number of studies	Number of techniques
14	1

8	1
5	2
4	1
3	2
2	7
1	14
Total	**28**

Table 6 lists the different publication fora in which the articles have been published. It is worth noting regarding the publication fora, that the empirical regression testing papers are published in a wide variety of journals and conference proceedings. Limiting the search to fewer journals and proceedings would have missed many papers, see Table 6.

The major software engineering journals and conferences are represented among the fora. It is not surprising that a conference on software maintenance is on the top, but we found, during the validity analysis, that the International Symposium on Software Testing and Analysis is not on the list at all. We checked the proceedings specifically and have also noticed that, for testing in general, empirical studies have been published there, as reported by Do *et al*. [8], but apparently not on regression test selection during the studied time period.

Table 6. Number of papers in different publication fora

Publication Fora	Type	#	%
International Conference on Software Maintenance	Conference	5	18.5
ACM Transactions of Software Engineering and Methodology	Journal	3	11.1
International Symposium on Software Reliability Engineering	Conference	3	11.1
International Conference on Software Engineering	Conference	3	11.1
Asia-Pacific Software Engineering Conference	Conference	2	7.4
International Symposium on Empirical Software Engineering	Conference	2	7.4
IEEE Transactions of Software Engineering	Journal	1	3.7
Journal of Systems and Software	Journal	1	3.7
Software Testing Verification and Reliability	Journal	1	3.7
Journal of Software Maintenance and Evolution	Journal	1	3.7
ACM SIGSOFT Symposium on Foundations of SE	Conference	1	3.7
Automated Software Engineering	Conference	1	3.7
Australian SE Conference	Conference	1	3.7
International Conf on COTS-based Software Systems	Conference	1	3.7
Int. Conference on Object-Oriented Programming, Systems, Languages, and Applications	Conference	1	3.7
Total		**27**	**100**

Table 7 lists authors with more than one publication. In addition to these 17 authors, five researchers have authored or co-authored one paper each. In the top of the author's list, we find the names of the most prolific researchers in the field of regression test selection (Rothermel and Harrold). It is interesting to notice from the point of view of conducting empirical software engineering research, that there are two authors on the top list with industry affiliation (Robinson and Smiley).

Table 7. Researchers and number of publications

Name	#
Rothermel G.	9
Harrold M. J.	5
Robinson B.	5
Zheng J.	4
Elbaum S. G.	3
Kallakuri P.	3
Malishevsky A.	3
Smiley K.	3
Williams L.	3

Name	#
Baradhi G.	2
Frankl P. G.	2
Kim J. M.	2
Mansour N.	2
Orso A.	2
Porter A.	2
White L.	2
Vokolos F.	2

The regression test selection techniques have been published from 1988 to 2006, as shown in Figure 2 and Table 8. The first empirical evaluations were published in 1997 (one case study and three experiments), hence the empirical evaluations have entered the scene relatively late. 12 out of the 28 techniques have been originally presented and evaluated in the same paper: T12-S11 and T13-S32 (1997); T14-S33-S34 (1999); T18-S5 (2003); T19-S13 (2004),; T20-S14; T21-S25; T23-S31; T25-S29-S30 and T26-S35 (2005); T27-S33 and T28-S35 (2006).

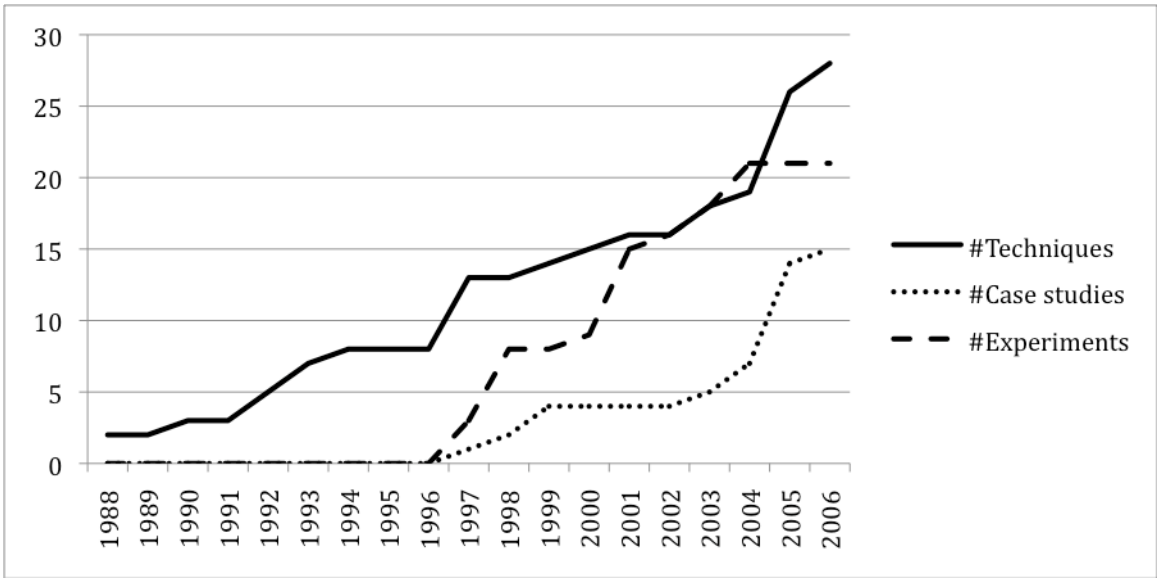


Figure 2. Accumulated number of published techniques, case studies and experiments.

We conclude from this analysis that there are only a few studies comparing many techniques in the same study, making it hard to find empirical data for a comprehensive comparison. However, some small and medium-sized artifacts have appeared as a de-facto benchmark in the field [8], enabling comparison to some extent of some techniques.

Most of the expected publication fora are represented, and one that is not represented, but was expected, was specifically double checked. Similarly, well known researchers in the field were among the authors, hence we consider the selected primary studies as being a valid set. It is clear from the publication analysis that the techniques published during the later years are published with empirical evaluations to a higher degree than during the earlier years, which is a positive trend in searching for empirically evaluated techniques as defined in RQ1.

3.3 Empirically evaluated techniques (RQ1)

3.3.1 Overview

Table 8 lists the 28 different regression test selection techniques (T1-T28), in chronological order according to date of first publication. In case the studies are reported partially or fully in different papers, we generally refer to the original one. In case a later publication has added details that are needed for exact specification of the technique, both references are used.

This list is specifically the answer to the first research question: which techniques for regression test selection existing in the literature have been evaluated empirically (RQ1). In this review, the techniques, their origin and description, are identified in accordance to what is stated in each of the selected papers, although adapted according to our definition of the what constitutes a unique technique in Section 2.5.

Table 8. Techniques for regression test selection

Technique	Origin	Description	Evaluated in study
T1	Harrold and Soffa (1988) [20]	Dataflow-coverage-based	S7
T2	Fischer et al. (1981) [13] Hartman and Robson (1988) [22]	Modification-focused, minimization, branch and bound algorithm	S5, S7, S9, S23, S24
T3	Leung and White (1990) [35]	Procedural-design firewall	S27, S29, S30
T4	Gupta et al. (1992) [16]	Coverage-focused, slicing	S1, S5, S11
T5	White and Leung (1992) [62]	Firewall	S1, S11
T6	Agraval et al. (1993)[1]	Incremental	S1, S11
T7	Rothermel and Harrold (1993)[47]	Viewing statements, DejaVu	S2 -S4, S6, S7, S9, S15 – S21, S31
T8	Chen and Rosenblum (1994) [6]	Modified entity - TestTube	S2 - S4, S9, S22 - 24
T9	Pei et al. (1997) [43] White and Abdullah (1997) [59]	High level – identifies changes at the class and interface level	S13, S25, S28 -S30

T10	Vokolos and Frankl (1997) [64]	Textual Differing - Pythia	S6, S26
T11	Mansour and Fakih (1997) [37]	Genetic algorithm	S1
T12	Mansour and Fakih (1997) [37]	Simulated annealing	S1, S11
T13	Wong et al. (1997) [66]	Hybrid: modification, minimization and prioritization- based selection	S32
T14	Wu et al. (1999) [67]	Analysis of program structure and function-calling sequences	S33, S34
T15	Rothermel et al. (2000) [51] Harrold et al. (2001) [19] Koju et al. (2003) [32]	Edge level - identifies changes at the edge level	S8, S10, S13
T16	Orso et al. (2001) [40]	Use of metadata to represent links between changes and Test Cases	S12
T17	Sajeev et al. (2003) [54]	Use of UML (OCL) to describe information changes	S12
T18	Elbaum et al. (2003) [11]	Modified-non-core Same as T8 but ignoring core functions	S5, S22
T19	Orso et al. (2004) [41]	Partitioning and selection Two Phases	S13
T20	Pasala and Bhowmick (2005) [42]	Runtime dependencies captured and modeled into a graph (CIG)	S14
T21	Skoglund and Runeson (2005) [56]	Change based selection	S25
T22	Willmor and Embury (2005)[63]	Test selection for DB-driven applications (extension of T7) combined safety	S31
T23	Willmor and Embury (2005) [63]	Database safety	S31
T24	Mao and Lu (2005) [38]	Enhanced representation of change information	S12
T25	White et al. (2005) [60]	Extended firewall additional data-paths	S29, S30
T26	Zheng (2005)[71]	I-BACCI v.1	S35
T27	Zheng et al. (2006)[74]	I-BACCI v.2 (firewall + BACCI)	S36
T28	Zheng et al. (2006) [74]	I-BACCI v.3	S35, S36
REF1	Leung and White (1989) [34]	Retest-all	S1 - S10, S12 – S24, S26, S31 - S36
REF2		Random (25)	S7, S9
REF3		Random (50)	S7, S9
REF4		Random (75)	S7, S9
REF5		Intuitive, experience based selection	S27, S28

3.3.2 **Development history**

The historical development chain gives some advice on which techniques are related and how they are developed, see Figure 3. There are three major paths, beginning with T3, T7 and T8 respectively.

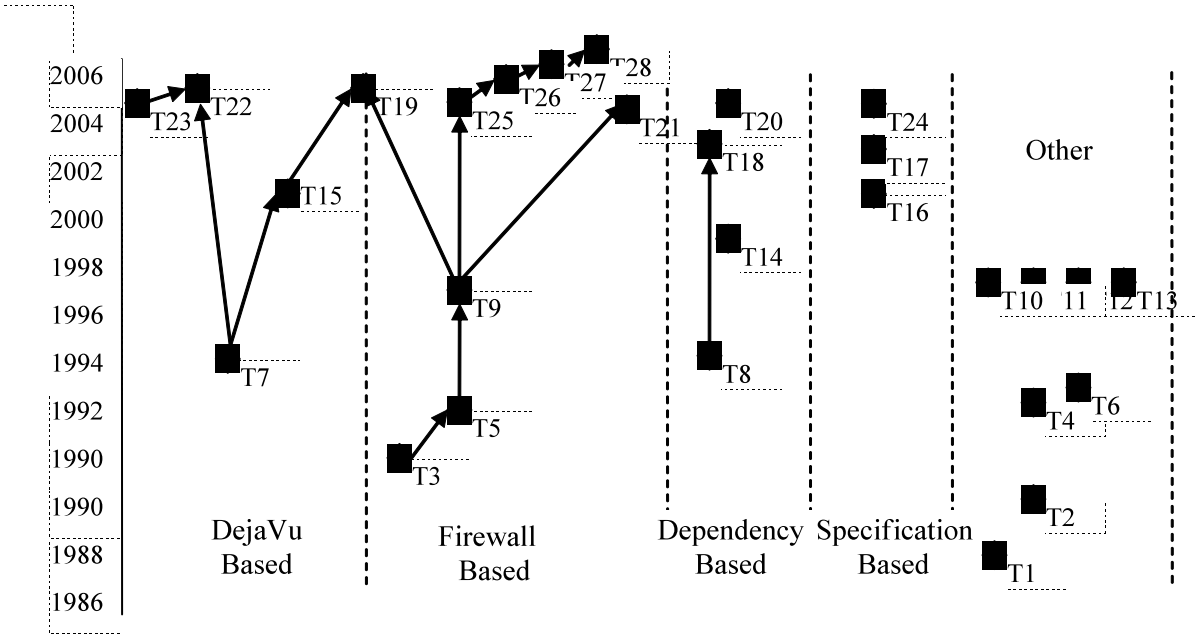


Figure 3. Evolution of techniques

One group of techniques is the *firewall* techniques where dependencies to modified software parts are isolated inside a firewall. Test cases covering the parts within the firewall are selected for re-execution. The first firewall technique (T3) for procedural languages was presented by Leung and White in 1990 [35]. An empirical evaluation used a changed version (T5). The technique was adapted to object-oriented languages T9 in two similar ways [43][59] and further enhanced and extended in the I-BACCI technique (T25-T28). It has also been adapted to Java (T21).

Another group of techniques is based on a technique invented by Rothermel and Harrold for procedural languages in 1993 [47] (T7), sometimes referred to as *DejaVu*. This technique has later been adopted to object-oriented languages T15 (for C++ [51], and for Java[19][32]) and also further extended for MSIL code [32]. Through technique T19 it has also been combined with techniques from the group of firewall techniques. Extended techniques that cope with database state have also been created T22 and T23 [63].

The firewall techniques are based on relationships to changed software parts. Different granularities of parts have been used, such as dependencies between modules, functions or classes. There exist techniques that are not stated in their presentations to be based on the firewall technique but still make use of dependencies between software parts. T8, T14 and T18 all utilize the relations between functions and T20 use dependencies between components (DLL:s).

In addition to the three major groups, there are other techniques which share some similarities with either group, although not being directly derived from one of them.

Using the dependency principle between larger parts, such as functions or classes, lead to that all test cases using the changed part are re-executed even though the actual modified code may not be executed. Using a smaller granularity gives better precision but are usually more costly since more analysis is needed. The smallest granularity is the program statements, segments, or blocks. The relationships between these smallest parts may be represented by creating control flow graphs where the control flow from one block to another may be seen as a relationship between the two blocks. This principle is for example used in the group of techniques based on Rothermel and Harrold's technique T7, see above, but is also used in the firewall technique T5. T10 also use program blocks for its test selection. An extension of this principle where the variables are also taken into account is used in the techniques T2, T4, T6, T11-T13, in various ways.

Another group of techniques are those using specifications or metadata of the software instead of the source code or executable code. T17 use UML specifications, and T16 and T24 use metadata in XML format for their test case selection.

3.3.3 Uniqueness of the techniques

There is a great variance regarding the uniqueness of the techniques identified in the studied papers. Some techniques may be regarded as novel at the time of their first presentation, while others may be regarded as only variants of already existing techniques. For example in [3] a regression test selection techniques is evaluated, T8, and the technique used is based on modified entities in the subject programs. In another evaluation, reported on in [11] it is stated that the same technique is used as in [3] but adapted to use a different scope of what parts of the subjects programs that is included in the analysis, T18. In [3] the complete subject programs are included in the analysis; while in [11] core functions of the subject programs are ignored. This difference of scope probably has an effect on the test cases selected using the two different approaches. The approach in which core functions is ignored is likely to select fewer test cases compared to the approach where all parts of the programs are included. It is not obvious whether the two approaches should be regarded as two different techniques or if they should be regarded as two very similar variants of the same technique. We chose the former option.

Some techniques evaluated in the reviewed papers are specified to be used for a specific type of software, e.g. Java, T15 and T19 [19][41], component based software, T17, T20, T24 and T28 [38][42][72][74], or database-driven applications, T22, [63]. It is not clear whether they should be considered one technique applied to two types of software, or two distinctly different techniques. For example, a technique specified for Java, T15, is presented and evaluated in [19]. In [58] the same technique is used on MSIL (MicroSoft Intermediate Language) code, however adapted to handle programming language constructs not present in Java. Thus, it can be argued that the results of the two studies cannot be synthesized in order to draw conclusions regarding the

performance of neither the technique presented in [19], nor the adapted version, used in [32]. However, we chose to classify them as the same technique.

There are also techniques specified in a somewhat abstract manner, e.g. techniques that handle object-oriented programs in general, e.g. T14 [67]. However, when evaluating a technique, the abstract specification of a technique must be concretized to handle the specific type of subjects selected for the evaluation. The concretization may look different depending on the programming language used for the subject programs. T14 is based on dependencies between functions in object-oriented programs in general. The technique is evaluated by first tailoring the abstract specification of the technique to C++ programs and then performing the evaluation on subject programs in C++. However, it is not clear how the tailoring of the specification should be performed to evaluate the technique using other object-oriented programming languages, e.g. C# or Java. Thus, due to differences between programming languages, a tailoring made for one specific programming language may have different general performance than a tailoring made for another programming language.

3.4 Classification of Techniques (RQ2)

In response to our second research question (RQ2), we are looking for some kind of classification of the regression test selection techniques. As indicated in Figure 3, there exist many variants of techniques, gradually evolved over time. Some suggested classifications of regression test techniques exist. Rothermel and Harrold present a framework for analyzing regression test selection techniques [48], including evaluation criteria for the techniques: inclusiveness, precision, efficiency and generality. Graves et al. [15] present a classification scheme where techniques are classified as Minimization, Safe, Dataflow-Coverage-based, Ad-hoc/Random or Retest-All techniques. Orso et al. [41] separate between techniques that operate at a higher granularity e.g. method or class (called high-level) and techniques that operate at a finer granularity, e.g. statements (called low-level). In this review we searched for classifications in the papers themselves with the goal of finding common properties in order to be able to reason about groups of regression testing techniques.

One property found regards the type of input required by the techniques. The most common type of required input is source code text, e.g. T1-8, T10-12 and T18. Other types of code analyzed by techniques are intermediate code for virtual machines, e.g. T9, T13-15 and T21, or machine code, e.g. T24 and T26**.** Some techniques require input of a certain format, e.g. T16 (meta data) and T17 (OCL). Techniques may also be classified according to the type of code used in the analysis (Java, C++…). A third type of classification that could be extracted from the papers regards the programming language paradigm. Some techniques are specified for use with procedural code, e.g. T1, T2, T7, T8, and T18, while other techniques are specified for the object-oriented paradigm, e.g. T9, T13-17, and T21-T23 some techniques are independent of programming language, e.g. T3, T19, and T26-28.

The most found property assigned to regression test selection techniques is whether they are *safe* or *unsafe*. With a safe technique the defects found with the full test suite are also found with the test cases picked by the regression test selection technique. This property may be used to classify all regression test selection techniques into either *safe* or *unsafe* techniques. Re-test all is an example of a safe technique since it selects all test cases, hence, it is guaranteed that all test cases that reveal defects are selected. Random selection of test cases is an example of an unsafe technique since there is a risk of test cases revealing defects being missed. In our study seven techniques were stated by the authors to be safe, T7, T8, T10, T15, and T21-24. However, the safety characteristic is hard to achieve in practice, as it e.g. assumes determinism in program and test execution.

A major problem, in addition to finding a classification scheme is applying the scheme to the techniques. The information regarding the different properties is usually not available in the publications. Hence, we may only give examples of techniques having the properties above based on what the authors state in their publications. The properties reported for each technique is presented in Table 9.

Table 9. Overview of properties for each technique.

	Applicability		Method			Properties	
Technique	Type of Language	Type of Software	Input	Approach	Granularity	Detection Ability	Cost Reduction
T1	Ind		IM	CF	Stm		
T2	Proc		SC	CF	Stm		Min
T3	Proc		SC	FW	Module		
T4	Proc		SC	Slicing	Stm		Min
T5	Proc		SC	FW	Module		
T6	Proc		SC	Slicing	Stm		
T7	Proc		SC	CF	Stm	Safe	
T8	Proc		SC	Dep	Func	Safe	
T9	OO		IM	FW	Class		
T10	Proc		SC		Stm	Safe	
T11	Proc		SC	CF	Stm		
T12	Proc		SC	CF	Stm		
T13	Proc		SC		Stm		Min
T14	OO		SC	Dep	Func		
T15	OO		IM	CF	Stm	Safe	
T16	OO	Comp	Spec	CF	Stm		
T17	OO	Comp	Spec				
T18	Proc		SC	Dep	Func		
T19	OO		IM	FW+CF	Class+Stm		
T20	Ind	Comp	BIN	Dep	Comp		
T21	OO		IM	FW	Class		
T22	OO	DB	SC	CF	Stm	Safe	
T23	OO	DB	SC	CF	Stm	Safe[2]	

[2] Safe only in DB-state

T24	OO	Comp	BIN +Spec	Dep	Stm	Safe	
T25	OO		SC?	FW	Class		
T26	Ind	Comp	BIN	FW	Func		
T27	Ind	Comp	BIN+SC	FW	Func		
T28	Ind	Comp	BIN+SC	FW	Func		
	Proc= Procedural language Ind = Independent OO = Object oriented	Comp = Component based DB = Database driven	SC = Source code IM = Intermediate code for virtual machines BIN = Machine code Spec = Input of a certain format	CF = Control flow FW = Fire wall Slicing Dependency based	Stm = statement Func = Function Class Module Component	Safe	Min = Minimization

3.5 Analysis of the Empirical Evidence (RQ3)

Once we have defined which empirical studies exist and a list of the techniques they evaluate, we continue with the third research question on whether there are significant differences between the techniques (RQ3). We give an overview of the primary studies as such in Subsection 3.5.1. Then we focus on the metrics and evaluation criteria used in different studies (3.5.2).

3.5.1 Types of empirical evidence

Table 10 overviews the primary studies by research method, and the size of the system used as subject. We identified 21 unique controlled experiments and 15 unique case studies. Half of the experiments are conducted on the same set of small programs [23], often referred to as the Siemens programs, which are made available through the software infrastructure repository[3] presented by Do *et al.* [8]. The number of large scale real life evaluations is sparse. In this systematic review we found four (S25, S27, S28, S30). Both types of studies have benefits and encounter problems, and it would be of interest to study the link between them, i.e. does a technique which is shown to have great advantages in a small controlled experiment show the same advantages in a large scale case study. Unfortunately no complete link was found in this review. However, the move from small toy programs to medium sized components, which is observed among the studies, is a substantial step in the right direction towards real-world relevance and applicability.

[3] http://sir.unl.edu

Table 10. Primary studies of different type and size

Type of studies	Size of subjects under study	Number of studies	%
Experiment	Large	1	3
Experiment	Medium	7	19
Experiment	Small	13	36
Case study	Large	4	11
Case study	Medium	5	14
Case study	Small	4	11
Case study	Not reported	2	6
	Total	**36**	**100**

The empirical quality of the studies varies a lot. In order to obtain a sufficiently large amount of papers, our inclusion criteria regarding quality had to be weak. Included in our analysis was any empirical evaluation of regression test selection techniques if relevant metrics were used and a sufficiently rigorous data collection and analysis could be followed in the report, see 2.4 for more details. This was independently assessed by two researchers.

An overview of the empirically studied relations between techniques and studies are shown in Figure 4. Circles represent techniques and connective lines between the techniques represent comparative studies. CS on the lines refers to the number of case studies conducted in which the techniques are compared, and Exp denotes the number of experimental comparisons. Some techniques have not been compared to any of the other techniques in the diagram: T13, T14 and T20. These techniques are still empirically evaluated in at least one study, typically a large scale case study. If no comparison between proposed techniques is made, the techniques are compared to a reference technique instead, e.g. the retest of all test cases, and in some cases a random selection of a certain percentage of test cases is used as a reference as well. The reference techniques are not shown in Figure 4 for visibility reasons.

Researchers are more apt to evaluate new techniques or variants of techniques than to replicate studies, which is clearly indicated by that we identified 28 different techniques in 27 papers. This gives rise to clusters of similar techniques compared among them selves and techniques only compared to a reference method such as re-test all.

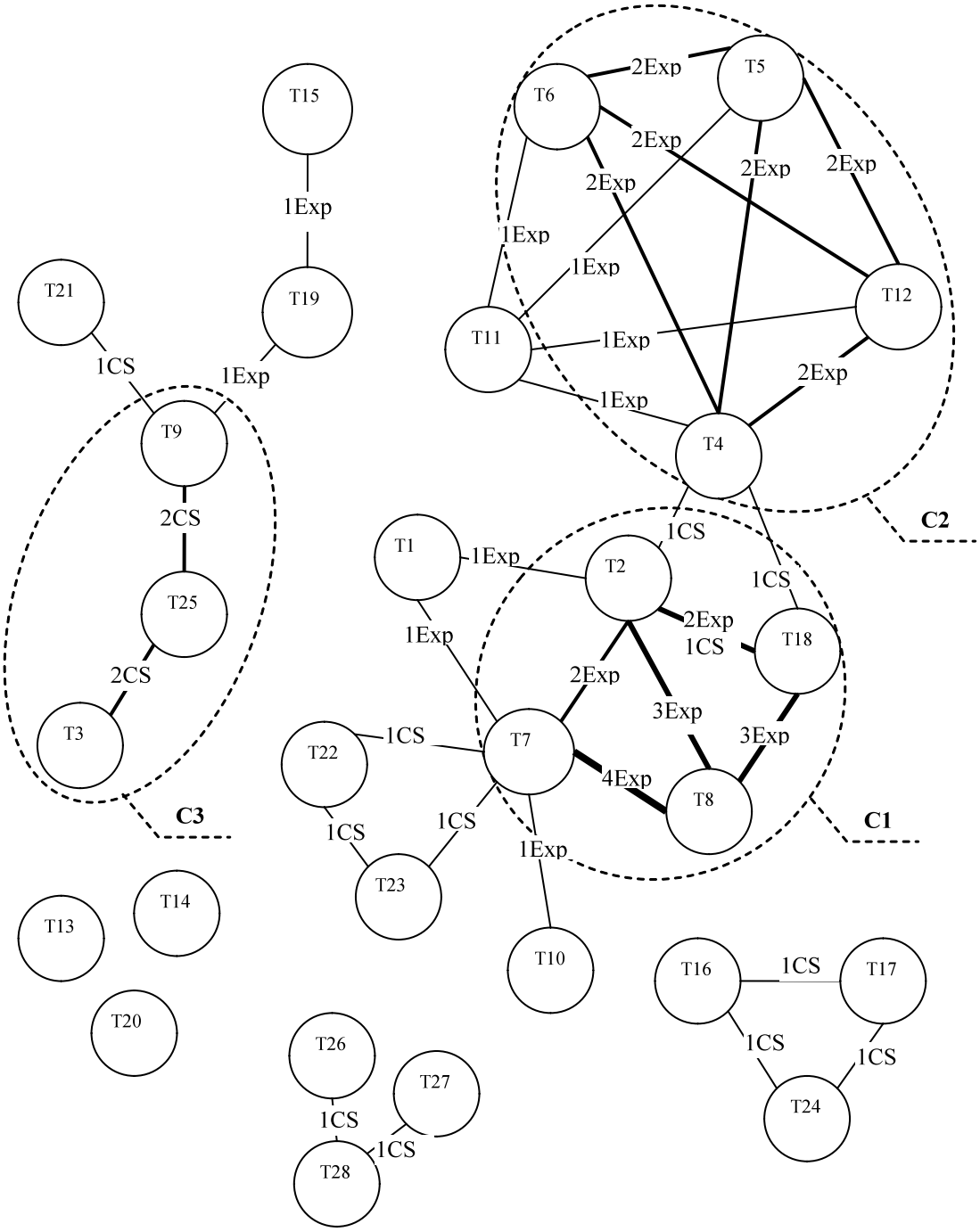


Figure 4. Techniques related to each other through empirical comparisons

Three clusters of techniques have been evaluated sufficiently to allow for meaningful comparison, see Figure 4; C1: T2, T7, T8 and T18, C2: T4, T5, T6 and T12, and C3: T3, T9 and T25. Each of these pair of techniques has been compared in at least two empirical studies. However, not all studies are conducted according to the same evaluation criteria, nor is the quality of the empirical evidence equally high. Therefore we classified the results with respect to empirical quality, as described in Section 2.6, and with respect to evaluation criteria, as desribed below.

3.5.2 Evaluation criteria

Do and Rothermel proposed a cost model for regression testing evaluation [9]. However, this model requires several data which is not published in the primary studies. Instead, we evaluated the results with respect to each evaluation criterion separately. We identified two main categories of metrics: *cost reduction* and *fault detection effectiveness*. Five different aspects of cost reduction and two of fault detection effectiveness have been evaluated in the primary studies. Table 11 gives an overview of the extent to which the different metrics are used in the studies. Size of test suite reduction is the most frequent, evaluated in 76% of the studies. Despite this, it may not be the most important metric. If the cost for performing the selection is too large in relation to this reduction, no savings are achieved. In 42% of the studies the total time (test selection and execution) is evaluated instead or as well. The effectiveness measures are either related 1) to test cases, i.e. the percentage of fault-revealing test cases selected out of all fault-revealing test cases, or 2) to faults, i.e. the percentage of faults out of all known ones, detected by the selected test cases.

Several of the studies concerning reduction of number of test cases are only compared to retest all (S8, S10, S14-S21, S26, S32-S34)[19][32][42][49][50][65][66][67] with the only conclusion that a reduction of test cases can be achieved, but nothing on the size of the effect in practice. This is a problem identified in experimental studies in general [26]. Many studies evaluating time reduction are conducted on small programs, and the size of the differences is measured in milliseconds, although there is a positive trend, over time, towards using medium-sized programs. Only 30% of the studies consider both fault detection and cost reduction. Rothermel proposed a framework for evaluation of regression test selection techniques [48] which have been used in some evaluations. This framework defines four key metrics, *inclusiveness, precision, efficiency*, and *generality*. Inclusiveness and precision corresponds to test case-related fault detection effectiveness and precision, respectively, in Table 11. Efficiency is related to space and time requirements and varies with test suite reduction as well as with test execution time and test selection time. Generality is more of a theoretical reasoning, which is not mirrored in the primary studies.

Table 11. Use of evaluation metrics in the studies

	Evaluated Metrics	**Number**	**%**	**Rothermel framework [48]**
Cost Reduction	Test suite reduction	29	76	Efficiency
	Test execution time	7	18	Efficiency
	Test selection time	5	13	Efficiency
	Total time	16	42	Efficiency
	Precision (omission of non-fault revealing tests)	1	3	Precision
Fault Detection Effectiveness	Test case-related detection effectiveness	5	13	Inclusiveness
	Fault-related detection effectiveness	8	21	

3.6 Comparison of Techniques (RQ43)

In response to our fourth research question (RQ4) we are analyzing the empirically evaluated relations between the techniques by visualizing the results of the studies. Due to the diversity in evaluation criteria and in empirical quality this visualization cannot give a complete picture. However, it may provide answers to specific questions: e.g. Is there any technique applicable in my context proven to reduce testing costs more than the one we use today?

Our taxonomy for analyzing the evidence follows the definitions in Table 2. Grey arrows indicate *light weight* empirical result and black arrows indicate *medium weight* result. A connection without arrows in the figures means that the studies have similar effect, while where there is a difference, the arrow points to the technique that is better with respect to the chosen criterion. A connection with thicker line represents more studies. In section 3.6.1, we report our findings regarding cost reduction and in section 3.6.2 regarding fault detection. Note that the numbers on the arrows indicate number of collected metrics, which may be more than one per study.

3.6.1 Cost reduction

Figure 5 reports the empirically evaluated relations between the techniques regarding the cost reduction, including evaluations of execution time as well as of test suite reduction and precision.

The strongest evidence can be found in cluster C1, where T2 provides most reduction of execution costs. T7, T8 and T18 reduce the test suites less than T2, and T8 among those reduces execution cost less than T18. All techniques however, reduce test execution cost compared to REF1 (re-test all), which is a natural criterion for a regression test selection technique.

In cluster C2, there is strong evidence that T6 and T12 have similar cost for test execution. On the other hand, there is a study with weaker empirical evidence, indicating that T12 reduces execution cost more than T6.

The rest of the studies show rather weak empirical evidence, showing that the evaluated techniques reduce test execution cost better than re-test all.

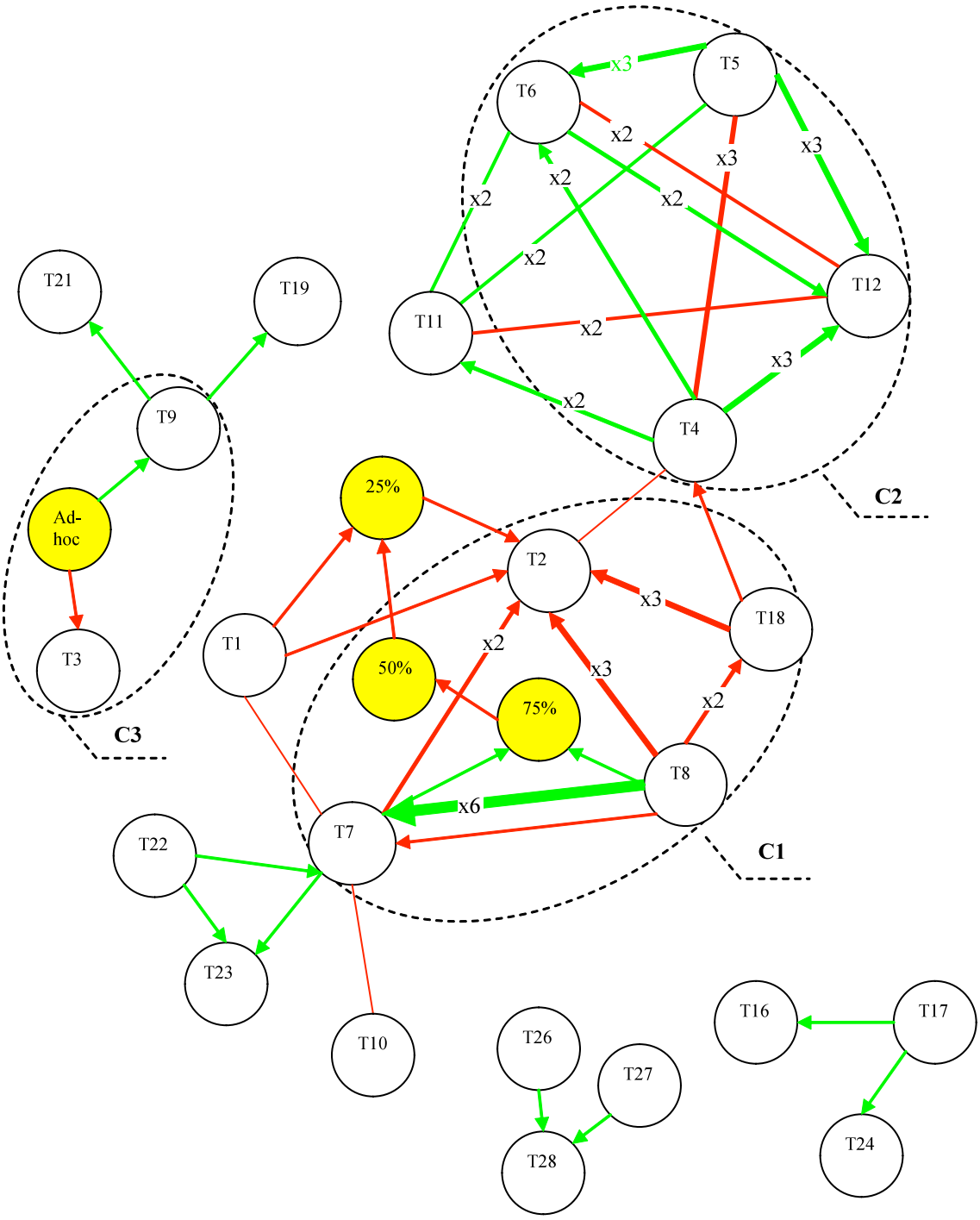


Figure 5. Empirical results for Cost Reduction, including Test Execution Time, Test Suite Reduction and Precision.

One component of the cost for regression test selection is the analysis time needed to select which test cases to re-execute. The selection time is reported separately for a small subset of the studies, as shown in Figure 6.

The left group primarily tells that T19 has less selection time than T15, and in C1, T8 has less analysis time than T7.

The results from cluster C2 shows mixed messages. T4 has in most cases the shortest selection time, although it in one study is more time consuming than T6. The selection time is hence dependent on the subject programs, test cases and types of changes done.

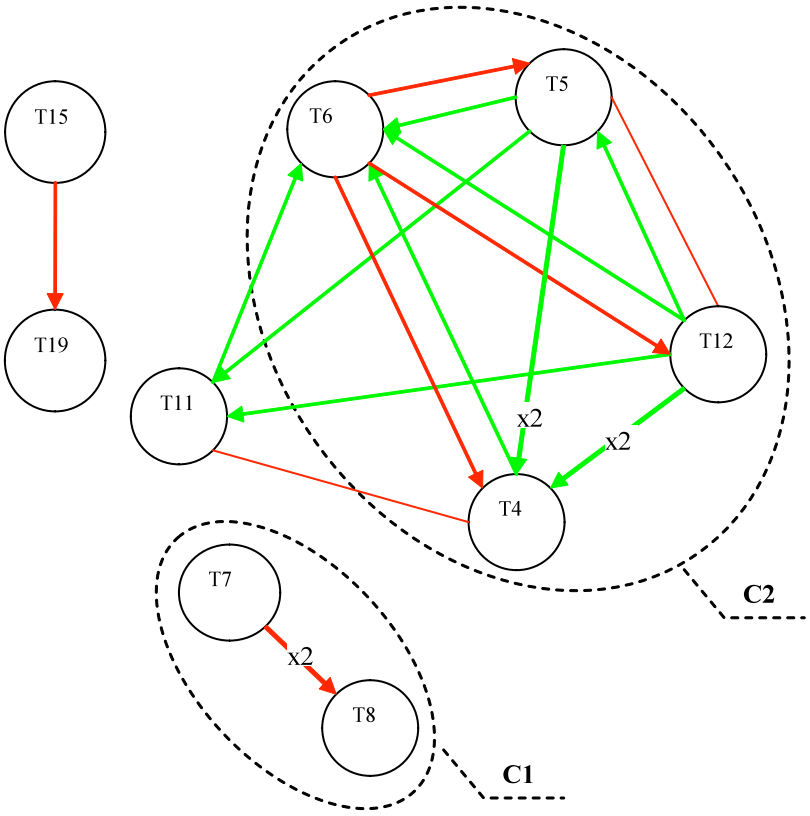


Figure 6. Empirical results for Test Selection Time

In Figure 7, the total time for analysis and execution together is shown for those studies where it is reported. It is worth noting that some regression test selection techniques actually can be more time consuming than re-test all (T7, T8, T10). Again, this is case dependent, but it is interesting to observe that this situation actually arises under certain conditions.

Other relations are a natural consequence of the expansion of certain techniques. T9 (Object oriented firewall) is less time consuming than T25 (extended OO firewall with data paths). Here an additional analysis is conducted in the regression test selection.

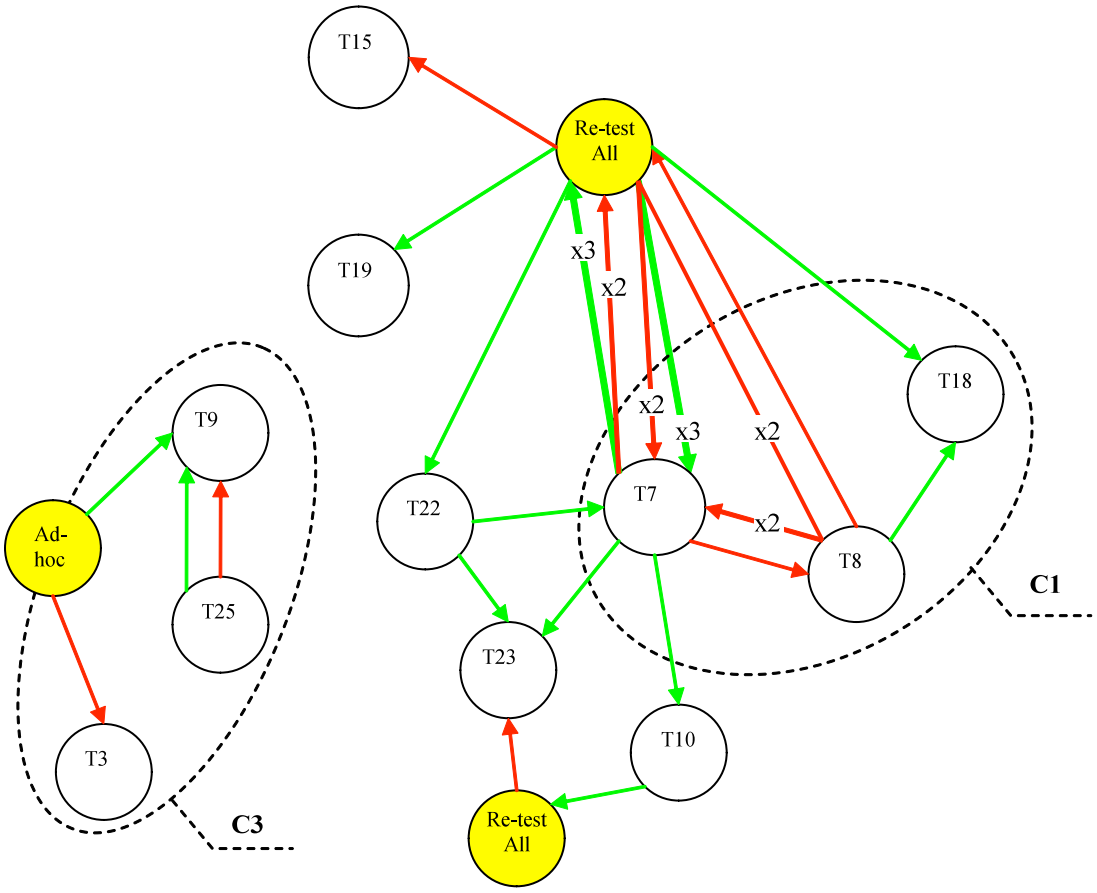


Figure 7. Empirical results for Total Time

3.6.2 Fault detection effectiveness

In addition to saving costs, regression test selection techniques should detect as many as possible of the faults found by the original test suite. Evaluations of test case-related as well as fault-related detection effectiveness are presented Figure 8.

Some techniques are proven to be *safe*, i.e. guarantees that the fault detection effectiveness is 100% compared to the original test suite (see Section 3.4). This property is stated to hold for seven techniques: T7, T8, T10, T15, T22, T23 and T24.

T7 and T8 within C2 are also those that can be found superior or equal from Figure 8, which is in line with the *safe* property. T4 in C2 tends also to be better or equal to all its reference techniques. However, for the rest, the picture is not clear.

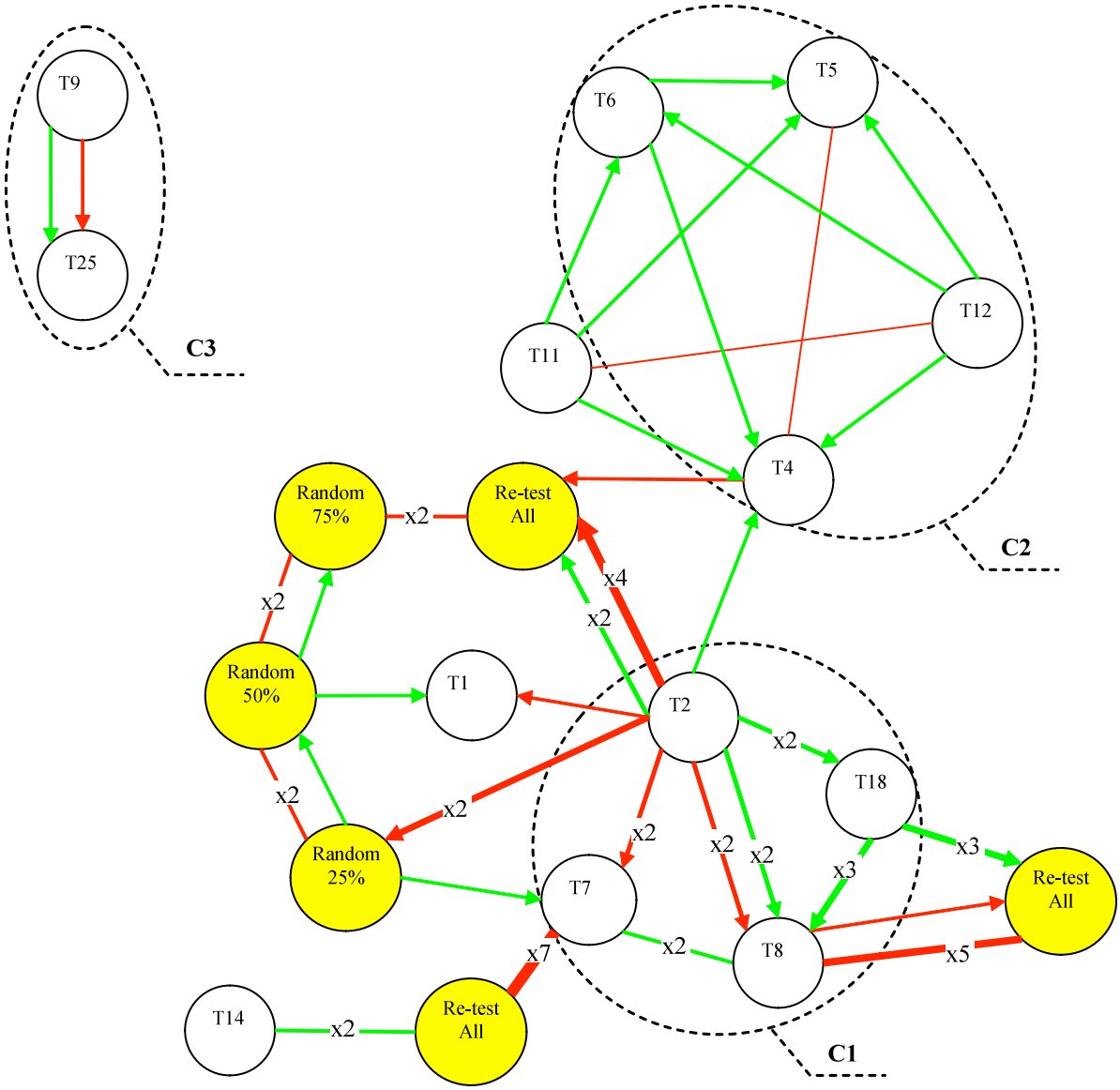


Figure 8. Empirical results for Fault Detection Effectiveness

4 DISCUSSION

4.1 The reviewed studies

The overall goal with the study was to identify regression test selection techniques and systematically assess the empirical evidence collected about those techniques. As the selection of a specific technique is dependent on many factors, the outcomes of empirical studies also depend on those factors. However only few factors are specifically addressed in the empirical studies and hence it is not possible to draw very precise conclusions. Nor is it possible to draw general conclusions. Instead we have conducted mostly qualitative assessments of the empirical studies. From those we try to aggregate recommendations of which regression test selection techniques to use.

A comparison of the techniques in cluster C1 indicates that the minimization technique, T2, is the most efficient in reducing time and/or number of test cases to run. However this is an unsafe technique (see Section 3.4) and all but one of six studies report on significant losses in fault detection. When it comes to safe techniques, T7 is shown to be the most efficient in reducing test cases. However analysis time for T7 is shown to be too long (it exceeds the time for rerunning all test cases) in early experiments, while in later experiments, it is shown to be good. Hence, there is a trade-off between cost reduction and defect detection ability. This is the case in all test selection, and none of the evaluated technique seems to have done any major breakthrough in solving this trade-off.

It is interesting to notice that the technique T7 is not changed between the studies that show different results on selection time, but the subject programs on which the experiments are conducted are changed. The subject programs is one factor that heavily impacts on the performance of some techniques. This emphasizes the importance of the regression testing context in empirical studies, and may also imply that specific studies have to be conducted when selecting a technique for a specific environment.

As mentioned before, many techniques are incremental improvements of existing techniques, which are demonstrated to perform better. For example, T25 is an extension of T9, with better fault detection at the cost of total time. This is a pattern shown in many of the studies: improvements may be reached, but always at a price for something else.

4.2 Implications for future studies

The standards for conducting empirical studies, and which measures to evaluate, differ greatly across the studies. Rothermel and Harrold proposed a framework to constitute the basis for comparison [48], but it is not used to any significant level in later research. Hence, it is not possible to conduct very strict aggregation of research results, e.g. through meta analysis. It is however not necessarily the ultimate goal to compare specific techniques. More general concepts would be more relevant to analyze, rather than detailed implementation issues.

Examples of such concepts to evaluate are indicated in the headings of Table 9. *Applicability*: are different techniques better suited for different languages or programming concepts, or for certain types of software? *Method*: are some selection approaches better suited to find faults, independently of details in their implementation? Which level of granularity for the analysis is effective – statement, class, component, or even specification level? Other concepts are related to process, product and resources factors [53]. *Process*: How frequent should the regression testing cycles be? At which testing level is the regression testing most efficient:

unit, function, system? *Product*: Is regression testing different for different types and sizes of products? *Resources*: Is the regression testing different with different skills and knowledge among the testers?

In the reviewed studies, some of these aspects are addressed: e.g. the size aspect, scaling up from small programs to medium-sized [50], the level of granularity of the change analysis [3], as well as testing frequency [27] and the effect of changes [11]. However, this has to be conducted more systematically by the research community.

Since the outcomes of the studies depend on many different factors, replication of studies with an attempt to keep as many factors stable as possible is a means to achieve a better empirical foundation for evaluation of concepts and techniques. The use of benchmarking software and test suites is one way of keeping factors stable between studies [8] However, in general, the strive for novelty in each research contribution tends to lead to a lack of replications and thus a lack of deeper understanding of earlier proposed techniques.

A major issue in this review is to find the relevant information to compare techniques. Hence, for the future, a more standardized documentation scheme would be helpful, as proposed by e.g. Jedlitschka and Pfahl [24]for experiments and Runeson and Höst [52] for case studies. To allow enough detail despite page restrictions, complementary technical reports could be published on the empirical studies.

5 CONCLUSIONS AND FUTURE WORK

In this paper we present results from a systematic review of empirical evaluations of regression test selection techniques. Related to our research questions we have identified that:

RQ1, there are 28 empirically evaluated techniques on regression test selection published,

RQ2. these techniques might be classified according to: applicability on type of software and type of language; details regarding the method such as which input is required, which approach is taken and on which level of granularity is changes considered; and properties such as classification in safe/unsafe or minimizing/not minimizing.

RQ3. the empirical evidence for differences between the techniques is not very strong, and sometimes contradictory, and

RQ4. hence there is no basis for selecting one superior technique. Instead techniques have to be tailored to specific situations, e.g. initially based on the classification of techniques.

We have identified some basic problems in the regression testing field which hinders a systematic review of the studies. Firstly, there is a great variance in the uniqueness of the techniques identified. Some techniques may be presented as novel at the time of their publications and others may be regarded as variants of already existing

techniques. Combined with a tendency to consider replications as second class research, the case for cooperative learning on regression testing techniques is not good. In addition to this, some techniques are presented in a rather general manner, e.g. claimed to handle object-oriented programs, which gives much space for different interpretations on how they may be implemented due to e.g. different programming language constructs existing in different programming languages. This may lead to different (but similar) implementations of a specific technique in different studies depending on e.g. the programming languages used in the studies.

As mentioned in Section 1, to be able to select a strategy for regression testing, relevant empirical comparisons between different methods are required. Where such empirical comparisons exist, the quality of the evaluations must be considered. One goal of this study was to determine whether the literature on regression test selection techniques provides such uniform and rigorous base of empirical evidence on the topic that makes it possible to use it as a base for selecting a regression test selection method for a given software system.

Our study shows that most of the presented techniques are not evaluated sufficiently for a practitioner to make decisions based on research alone. In many studies, only one aspect of the problem is evaluated and the context is too specific to be easily applied directly by software developers. Few studies are replicated, and thus the possibility to draw conclusions based on variations in test context is limited. Of course even a limited evidence base could be used as guidance. In order for a practitioner to make use of these results, the study context must be considered and compared to the actual environment into which a technique is supposed to be applied.

Future work for the research community is 1) focus more on general regression testing concepts rather than on variants of specific techniques; 2) encourage systematic replications of studies in different context, preferably with a focus on gradually scaling up to more complex environments; 3) define how empirical evaluations of regression test selection techniques should be reported, which variation factors in the study context are important.

6 ACKNOWLEDGMENTS

The authors acknowledge Dr. Carina Andersson for her contribution to the first two stages of the study. The authors are thankful to librarian Maria Johnsson for excellent support in the search procedures. We appreciate review comments from Prof. Sebastian Elbaum and the anonymous reviewers, which substantially have improved the paper. The work is partly funded by the Swedish Governmental Agency for Innovation Systems under grant 2005-02483 for the UPPREPA project, and partly funded by the Swedish Research Council under grant 622-2004-552 for a senior researcher position in software engineering.

7 REFERENCES

[1] Agrawal, H., Horgan, J.R., Krauser, E.W., and London, S.A. 1993. Incremental regression testing. In Proceedings. Conference on Software Maintenance 1993. CSM-93 (Cat. No.93CH3360-5). IEEE Comput. Soc. Press, 348-57.

[2] Baradhi, G. and Mansour, N. 1997. A comparative study of five regression testing algorithms. Software Engineering Conference, 1997. Proceedings. 1997 Australian. 174-182.

[3] Bible, J., Rothermel, G., and Rosenblum, D.S. 2001. A comparative study of coarse- and fine-grained safe regression test-selection techniques. ACM Transactions on Software Engineering and Methodology. 10(2), 149-183.

[4] Binkley, D. 1998. The application of program slicing to regression testing. Information and Software Technology. 40(11-12), 583-94.

[5] Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., and Khalil, M. 2007. Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software. 80(4), 571-83.

[6] Chen, Y.-F., Rosenblum, D.S., and Vo, K.-P. 1994. Test tube: A system for selective regression testing. In Proceedings - International Conference on Software Engineering. IEEE, Los Alamitos, CA, USA, 211-220.

[7] Dieste, O., Grimán, A., and Juristo, N. 2008. Developing search strategies for detecting relevant experiments. Empirical Software Engineering.

[8] Do, H. Elbaum, S. and Rothermel, G. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact, Empirical Software Engineering, An International Journal, V. 10, No. 4, October 2005

[9] Do, H. and Rothermel, G., An empirical study of regression testing techniques incorporating context and lifecycle factors and improved cost-benefit models, Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering, November 2006, pages 141-151.

[10] Dybå, T., Dingsöyr, T., and Hanssen, G.K. 2007. Applying Systematic Reviews to Diverse Study Types: An Experience Report. In First International Symposium on Empirical Software Engineering and Measurement, 2007, ESEM 2007. 225-234.

[11] Elbaum, S., Kallakuri, P., Malishevsky, A., Rothermel, G., and Kanduri, S. 2003. Understanding the effects of changes on the cost-effectiveness of regression testing techniques. Software Testing, Verification and Reliability. 13(2), 65-83.

[12] Engström, E., Skoglund, Mats, Runeson, Per. 2008. Empirical Evaluations of Regression Test Selection Techniques: A Systematic Review. ESEM 08

[13] Fischer, K., Raji, F., and Chruscicki, A. 1981. A methodology for retesting modified software. In NTC '81. IEEE 1981 National Telecommunications Conference. Innovative Telecommunications - Key to the Future. IEEE, 6-3.

[14] Frankl, P.G., Rothermel, G., Sayre, K., and Vokolos, F.I. 2003. An empirical comparison of two safe regression test selection techniques. Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on. 195-204.

[15] Graves, T.L., Harrold, M.J., Kim, J.M., Porter, A., and Rothermel, G. 2001. An empirical study of regression test selection techniques. ACM Transactions on Software Engineering and Methodology. 10(2), 184-208.

[16] Gupta, R., Harrold, M.J., and Soffa, M.L. 1992. An approach to regression testing using slicing. In Conference on Software Maintenance 1992 (Cat.No.92CH3206-0). IEEE Comput. Soc. Press, 299-308.

[17] Gupta, R., Harrold, M.J., and Soffa, M.L. 1996. Program slicing-based regression testing techniques. Software Testing, Verification and Reliability. 6(2), 83-111.

[18] Haftmann, F., Kossmann, D., and Lo, E. 2007. A framework for efficient regression tests on database applications. VLDB Journal. 16(1), 145-64.

[19] Harrold, M.J., Jones, J.A., Tongyu, L., Donglin, L., Orso, A., Pennings, M., Sinha, S., Spoon, S.A., and Gujarathi, A. 2001. Regression test selection for Java software. In SIGPLAN Not. (USA). ACM, 312-26.

[20] Harrold, M.J. and Souffa, M.L. 1988. An incremental approach to unit testing during maintenance. In Proceedings of the Conference on Software Maintenance - 1988 (IEEE Cat. No.88CH2615-3). IEEE Comput. Soc. Press, 362-7.

[21] Hartmann, J. and Robson, D.J. 1988. Approaches to regression testing. In Proceedings of the Conference on Software Maintenance - 1988 (IEEE Cat. No.88CH2615-3). IEEE Comput. Soc. Press, 368-72.

[22] Hartmann, J. and Robson, D.J. 1990. Techniques for selective revalidation. IEEE Software. 7(1), 31-6.

[23] Hutchins, M., Foster, H., Goradia, T., and Ostrand, T. 1994. Experiments on the effectiveness of dataflow- and control-flow-based test adequacy criteria. In ICSE-16. 16th International Conference on Software Engineering (Cat. No.94CH3409-0). IEEE Comput. Soc. Press, 191-200.

[24] Jedlitschka, A.and Pfahl, D. 2005. Reporting Guidelines for Controlled Experiments in Software Engineering, In Proceedings of ACM/ IEEE International Symposium on Empirical Software Engineering, pp 95-104

[25] Juristo, N., Moreno, A.M., Vegas, S., and Solari, M. 2006. In search of what we experimentally know about unit testing [software testing]. IEEE Software. 23(6), 72-80.

[26] Kampenes Vigdis, B., Dybå, T., Hannay Jo, E., and Sjöberg Dag, I.K. 2007. A systematic review of effect size in software engineering experiments. Information and Software Technology. 49(11-12), 1073-1073.

[27] Kim, J.-M., Porter, A., and Rothermel, G. 2005. An empirical study of regression test application frequency. Software Testing, Verification and Reliability. 15(4), 257-279.

[28] Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. 2009. Systematic literature reviews in software engineering - A systematic literature review. Information and Software Technology. Volume 51(Issue 1), Pages 7-15.

[29] Kitchenham, B.A. 2007. Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3. Technical Report S.o.C.S.a.M. Software Engineering Group, Keele University and Department of Computer Science University of Durham.

[30] Kitchenham, B.A., Mendes, E., and Travassos, G.H. 2007. Cross versus within-company cost estimation studies: a systematic review. IEEE Transactions on Software Engineering. 33(5), 316-29.

[31] Klosch, R.R., Glaser, P.W., and Truschnegg, R.J. 2002. A testing approach for large system portfolios in industrial environments. Journal of Systems and Software. 62(1), 11-20.

[32] Koju, T., Takada, S., and Doi, N. 2003. Regression Test Selection based on Intermediate Code for Virtual Machines. In Conference on Software Maintenance. Institute of Electrical and Electronics Engineers Inc., 420-429.

[33] Landis, J.R. and Gary, G.K. 1977. The Measurement of Observer Agreement for Categorical Data. Biometrics. 33(1), 159-174.

[34] Leung, H.K.N. and White, L. 1990. Insights into testing and regression testing global variables. Journal of Software Maintenance: Research and Practice. 2(4), 209-22.

[35] Leung, H.K.N. and White, L. 1990. A study of integration testing and software regression at the integration level. In Proceedings. Conference on Software Maintenance 1990 (Cat. No.90CH2921-5). IEEE Comput. Soc. Press, 290-301.

[36] Mansour, N., Bahsoon, R., and Baradhi, G. 2001. Empirical comparison of regression test selection algorithms. The Journal of Systems and Software. 57(1), 79-90.

[37] Mansour, N. and El-Fakih, K. 1997. Natural optimization algorithms for optimal regression testing. In Proceedings - IEEE Computer Society's International Computer Software & Applications Conference. IEEE, Los Alamitos, CA, USA, 511-514.

[38] Mao, C. and Lu, Y. 2005. Regression testing for component-based software systems by enhancing change information. In Proceedings. 12th Asia-Pacific Software Engineering Conference. IEEE Computer Society, 8 pp.

[39] Memon, A.M. 2004. Using tasks to automate regression testing of GUIs. In IASTED International Conference on Artificial Intelligence and Applications - AIA 2004. ACTA Press, 477-82.

[40] Orso, A., Harrold, M.J., Rosenblum, D., Rothermel, G., Soffa, M.L., and Do, H. 2001. Using component metacontent to support the regression testing of component-based software. In Proceedings IEEE International Conference on Software Maintenance. ICSM 2001. IEEE Comput. Soc, 716-25.

[41] Orso, A., Nanjuan, S., and Harrold, M.J. 2004. Scaling regression testing to large software systems. In Softw. Eng. Notes (USA). ACM, 241-51.

[42] Pasala, A. and Bhowmick, A. 2005. An approach for test suite selection to validate applications on deployment of COTS upgrades. In Proceedings - Asia-Pacific Software Engineering Conference, APSEC. IEEE Computer Society, Los Alamitos, CA 90720-1314, United States, 401-407.

[43] Pei, H., Xiaolin, L., Kung, D.C., Chih-Tung, H., Liang, L., Toyoshima, Y., and Chen, C. 1997. A technique for the selective revalidation of OO software. Journal of Software Maintenance: Research and Practice. 9(4), 217-33.

[44] Ren, X., Shah, F., Tip, F., Ryder, B.G., and Chesley, O. 2004. Chianti: A tool for change impact analysis of java programs. In 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA'04. Association for Computing Machinery, New York, NY 10036-5701, United States, 432-448.

[45] Rothermel, G., Elbaum, S., Malishevsky, A., Kallakuri, P., and Davia, B. 2002. The impact of test suite granularity on the cost-effectiveness of regression testing. In Proceedings - International Conference on Software Engineering. Institute of Electrical and Electronics Engineers Computer Society, 130-140.

[46] Rothermel, G., Elbaum, S., Malishevsky, A.G., Kallakuri, P., and Xuemei, Q. 2004. On test suite composition and cost-effective regression testing. ACM Transactions on Software Engineering and Methodology. 13(3), 227-331.

[47] Rothermel, G. and Harrold, M.J. 1993. A safe, efficient algorithm for regression test selection. Software Maintenance ,1993. CSM-93, Proceedings., Conference on. 358-367.

[48] Rothermel, G. and Harrold, M.J. 1996. Analyzing regression test selection techniques. IEEE Transactions on Software Engineering. 22(8), 529-51.

[49] Rothermel, G. and Harrold, M.J. 1997. A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology. 6(2), 173-210.

[50] Rothermel, G. and Harrold, M.J. 1998. Empirical studies of a safe regression test selection technique. IEEE Transactions on Software Engineering. 24(6), 401-19.

[51] Rothermel, G., Harrold, M.J., and Dedhia, J. 2000. Regression test selection for C++ software. Journal of Software Testing Verification and Reliability. 10(2), 77-109.

[52] Runeson, P. and Höst, M. Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering, 14(2):131-164, 2009.

[53] Runeson, P., Skoglund, M. and Engström, E. Test Benchmarks – what is the question?, TestBench Workshop at International Conference on Software Testing, Verification and Validation, Lillehammer, Norway, April 2008.

[54] Sajeev, A.S.M. and Wibowo, B. 2003. Regression test selection based on version changes of components. In Tenth Asia-Pacific Software Engineering Conference. IEEE Comput. Soc, 78-85.

[55] Shadish, T., Cook, T., and Campbell, D. 2002. Experimental and Quasi-Experimental Designs - for Generalized Causal Inference. 2 ed, Boston: Houghton Mifflin Company. 623.

[56] Skoglund, M. and Runeson, P. 2005. A case study of the class firewall regression test selection technique on a large scale distributed software system. In 2005 International Symposium on Empirical Software Engineering (IEEE Cat. No. 05EX1213). IEEE, 10 pp.

[57] Staples, M. and Niazi, M. 2007. Experiences using systematic review guidelines. The Journal of Systems & Software. 80(9), 1425-37.

[58] Toshihiko, K., Shingo, T., and Norihisa, D. 2003. Regression test selection based on intermediate code for virtual machines. In Proceedings International Conference on Software Maintenance ICSM 2003. IEEE Comput. Soc, 420-9.

[59] White, L. and Abdullah, K. 1997. A firewall approach for the regression testing of object-oriented software. Software Quality Week

[60] White, L., Jaber, K., and Robinson, B. 2005. Utilization of extended firewall for object-oriented regression testing. In IEEE International Conference on Software Maintenance, ICSM. IEEE Computer Society, Los Alamitos, CA 90720-1314, United States, 695-698.

[61] White, L. and Robinson, B. 2004. Industrial real-time regression testing and analysis using firewalls. Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on. 18-27.

[62] White, L.J. and Leung, H.K.N. 1992. A firewall concept for both control-flow and data-flow in regression integration testing. In Conference on Software Maintenance 1992 (Cat.No.92CH3206-0). IEEE Comput. Soc. Press, 262-71.

[63] Willmor, D. and Embury, S.M. 2005. A safe regression test selection technique for database-driven applications. In Proceedings of the 21st IEEE International Conference on Software Maintenance. IEEE Comput. Soc, 421-30.

[64] Vokolos, F.I. and Frankl, P.G. 1997. Pythia: a regression test selection tool based on textual differencing. In Reliability, Quality and Safety of Software-Intensive Systems. IFIP TC5 WG5.4 3rd International Conference. Chapman & Hall, 3-21.

[65] Vokolos, F.I. and Frankl, P.G. 1998. Empirical evaluation of the textual differencing regression testing technique. In Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272). IEEE Comput. Soc, 44-53.

[66] Wong, W.E., Horgan, J.R., London, S., and Agrawal, H. 1997. A study of effective regression testing in practice. In Proceedings. The Eighth International Symposium on Software Reliability Engineering (Cat. No.97TB100170). IEEE Comput. Soc, 264-74.

[67] Wu, Y., Chen, M.-H., and Kao, H.M. 1999. Regression testing on object-oriented programs. In Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No.PR00443). IEEE Comput. Soc, 270-9.

[68] Yanping, C., Robert, L.P., and Sims, D.P. 2002. Specification-based regression test selection with risk analysis. Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research. IBM Press.

[69] Yin, R.K. 2003. Case Study Research - Design and Methods Applied Social Research Methods Series, ed. D.J.R. Leonard Bickman. Vol. 5, London: Sage Publications.

[70] Zheng, J. 2005. In regression testing selection when source code is not available. Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM.

[71] Zheng, J., Robinson, B., Williams, L., and Smiley, K. 2005. An initial study of a lightweight process for change identification and regression test selection when source code is not available. In Proceedings - International Symposium on Software Reliability Engineering, ISSRE. IEEE Computer Society, 225-234.

[72] Zheng, J., Robinson, B., Williams, L., and Smiley, K. 2006. Applying regression test selection for COTS-based applications. In Proceedings - International Conference on Software Engineering. Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 512-521.

[73] Zheng, J., Robinson, B., Williams, L., and Smiley, K. 2006. An initial study of a lightweight process for change identification and regression test selection when source code is not available. In Proceedings. 16th IEEE International Symposium on Software Reliability Engineering. IEEE Computer Society, 10 pp.

[74] Zheng, J., Robinson, B., Williams, L., and Smiley, K. 2006. A lightweight process for change identification and regression test selection in using COTS components. In Proceedings - Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems. Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 137-143.