Effective Regression Test Case Selection: A Systematic Literature Review

RAFAQUT KAZMI, DAYANG N. A. JAWAWI, and RADZIAH MOHAMAD,
Universiti Teknologi Malaysia
IMRAN GHANI, Monash University Malaysia

Regression test case selection techniques attempt to increase the testing effectiveness based on the measurement capabilities, such as cost, coverage, and fault detection. This systematic literature review presents state-of-the-art research in effective regression test case selection techniques. We examined 47 empirical studies published between 2007 and 2015. The selected studies are categorized according to the selection procedure, empirical study design, and adequacy criteria with respect to their effectiveness measurement capability and methods used to measure the validity of these results.

The results showed that mining and learning-based regression test case selection was reported in 39% of the studies, unit level testing was reported in 18% of the studies, and object-oriented environment (Java) was used in 26% of the studies. Structural faults, the most common target, was used in 55% of the studies. Overall, only 39% of the studies conducted followed experimental guidelines and are reproducible.

There are 7 different cost measures, 13 different coverage types, and 5 fault-detection metrics reported in these studies. It is also observed that 70% of the studies being analyzed used cost as the effectiveness measure compared to 31% that used fault-detection capability and 16% that used coverage.

CCS Concepts: • **Software and its engineering** → *Software testing and debugging*; *Empirical software validation*

Additional Key Words and Phrases: Software testing, SLR, coverage, cost effectiveness, fault detection ability

ACM Reference Format:
Rafaqut Kazmi, Dayang N. A. Jawawi, Radziah Mohamad, and Imran Ghani. 2017. Effective regression test case selection: A systematic literature review. ACM Comput. Surv. 50, 2, Article 29 (May 2017), 32 pages.
DOI: http://dx.doi.org/10.1145/3057269

1. INTRODUCTION

Regression testing is a repetitive part of software testing. It ensures that new defects will not be introduced into the extended code or specification changes. IEEE defines regression testing as (IEEE-Std-610.12-1990 1990):

> *"Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or components still complies with its specified requirement."*

The use of regression testing may increase due to the growth in iterative development as well as reusability of different software artifacts at different levels of software

Authors' addresses: R. Kazmi, D. N. A. Jawawi, R. Mohamad, Department of Software Engineering Faculty of Computing Universiti Teknologi Malaysia 81310 UTM Johor Bahru, Johor, Malaysia; emails: rafaqutkazmi@gmail.com, dayang@utm.my, radziahm@utm.my; I. Ghani, School of Information Technology, Monash University Malaysia, 47500 Bandar Sunway Selangor, Malaysia; email: imran.ghani@monash.edu.

DOI: http://dx.doi.org/10.1145/3057269

projects. This issue necessitates the need to select test cases effectively, which is a challenging task. This is because it affects the cost, coverage, and fault detection during regression testing. As mentioned in several studies, 50% of a project's cost is spent on software testing and 80% of this amount is consumed by regression testing [Chittimalli and Harrold 2009; Tao et al. 2010a; White 1989]. To maintain the effectiveness (cost, coverage, fault) of test suite in regression testing, tester may reduce size of test suite, number of test case execution, test case execution time, select a subset of test cases that have already been executed on the system under testing (SUT), or effectively prioritize the test cases [Askarunisa et al. 2010]. The behavior of regression test selection (RTS) and test suite reduction are almost similar. They both attempt to select a subset of test cases from previously executed test suites. But the main difference between them is RTS is based on code modifications in SUT and selects the test cases that are relevant to the modifications between the previous and current version. On the contrary, reduction methods rely on coverage information metrics for a single version. On the other side, test suite prioritization reorders test cases with the intent to discover faults from SUT as early as possible. The main difference between RTS and prioritization techniques is prioritization does not account for code modifications in SUT. The main focus of prioritization techniques is to increase fault detection ability, ignoring the version of SUT or modifications made to the source code of SUT [Yoo and Harman 2012]. Coverage information with fault detection is used as proxy for regression test case selection [Rothermel et al. 1998] and in other studies, code coverage information is also utilized for cost reduction in RTS techniques [Rosenblum and Weyuker 1997; Binkley 1995; Inozemtseva and Holmes 2014]. The scope chosen for this systematic literature review (SLR) is effectiveness of RTS. This scope is chosen because these techniques are related to a common objective of regression testing optimization. These techniques aim to optimize regression testing from an existing pool of test cases based on changes in the current version of the SUT.

These three parameters (cost, coverage, and fault detection capability) are chosen because measurement of effectiveness is context-dependent [Do and Rothermel 2006]. The context we mean is circumstances that form the existence of an event, which in current scenario is regression testing. The context of regression testing is different for different testing constraints, such as configuration aware regression testing [Qu et al. 2008], time aware regression testing [Zhang et al. 2009], faulty control flow path aware regression testing [Jiang and Su 2007], and fault-free time aware regression testing [Yoo and Harman 2012]. The measurement of software testing costs can be divided into two: direct measures and indirect measures [Leung and White 1991]. Direct measures include time for test specification, analysis, design, execution, and post analysis. Indirect costs include managerial cost, maintenance cost, and tools cost. Effectiveness is determined on the basis of coverage, fault detection, and direct costs. There is a close relationship between cost and effectiveness [Elbaum et al. 2003]. A test case is said to be effective if it finds more bugs [Orso and Rothermel 2014]. There are other ways to measure the effectiveness of regression testing as reported in the literature. Coverage is the main contributor of cost in regression testing with its fault detection [Rosenblum and Weyuker 1997]. Unfortunately, many researchers do not agree on the relationship between costs, coverage, and fault detection [Andrews et al. 2006; Cai and Lyu 2005; Namin and Andrews 2009; Gligoric et al. 2014]. Effectiveness is also measured in terms of effort reduction in creating and maintaining test suites [Poulding et al. 2007], coverage information of SUT [Tao et al. 2010b], and cost-effectiveness of test suite [Nagar et al. 2014]. There is a need to find the relationships between effectiveness contributors and its impact on RTS techniques.

Regression testing is used in different applications, such as database [Haftmann et al. 2007], graphical user interface (GUI) [Memon and Soffa 2003; Memon 2008; Memon

Table I. Summary of Related Studies in Regression Testing

Study Type	Study References	Study Focus	Year of Publication	Total Studies Reviewed	Years Covered
SLRs	[Engström et al. 2010]	Test Case Selection	2010	27	1988–2006
	[Engström et al. 2008]	Test Case Selection	2008	28	1969–2006
Surveys	[Orso and Rothermel 2014]	Software Testing	2014	—	2000–2014
	[Yoo and Harman 2012]	Regression Testing	2012	159	1977–2009
	[Biswas et al. 2011b]	Test Case Selection	2011	—	
Mapping	[González et al. 2014]	Software Testing	2014	16	2000–2013

et al. 2005], web [Xu et al. 2003; Kung et al. 2000; Harman and Alshahwan 2008; Roest et al. 2010], and real-time embedded systems applications [Schütz 1994; Hla et al. 2008; White and Robinson 2004]. In Biswas et al. [2011b], the authors compared different regression testing techniques. They also classify those techniques with respect to application types. This survey highlights the need to investigate the relationship between cost- and coverage-based effectiveness. There is not enough evidence found from SLRs, mapping studies, and surveys on the relationship between cost, coverage, and fault detection as shown in Table I.

The organization of this article is as follows. Section 2 discusses the background of empirical evidence for SLR RTS selection techniques. Section 3 describes a framework to evaluate the quality of empirical studies selected. Section 4 describes research method adopted to conduct this SLR. Section 5 represents and discusses the results. Section 6 describes the threats to validity with regard to this SLR. The discussion and conclusion for this SLR are presented in Section 7 and Section 8, respectively. Finally, Section 9 presents future works with regard to this SLR.

2. BACKGROUND OF REGRESSION TEST CASE SELECTION STUDIES

As mentioned in the Introduction, in this SLR, the authors have collected relevant articles discussing regression test case selection techniques that focused on cost-, coverage-, or fault-based effectiveness. In the literature, there are only two SLRs, one mapping study, and three surveys that are related to regression testing or regression test case selection, as shown in Table I. But these studies did not focus on effectiveness based on cost, coverage, and fault detection of regression test case selection. Let us discuss these studies one by one.

The first SLR [Engström et al. 2010] covered the time period between 1988 and 2006 and identified 27 studies, including 38 empirical studies, 21 experiments, and 25 case studies. The SLR identified 28 regression test case selection techniques and compared these techniques on the basis of cost reduction and fault detection capability. The second SLR [Engström et al. 2008] covered the time period between 1969 and 2006 and identified 28 primary studies for regression test case selection. The identified regression test case selection techniques are classified on the basis of their input method, type of code under analysis, and selection criteria. The authors of these SLRs concluded that there is a need to put research focus on medium to large data sets for empirical studies and there is also a need to investigate regression test case selection using better empirical designs and evaluation methods. The conclusion for these two SLRs were the same as presented in Engström et al. [2010].

In addition, there is one mapping study [González et al. 2014] found in the literature. It covered the time period between 2000 and 2013. The results are the same as reported in previous SLRs with the same shortcomings. The authors identified four testing families on the basis of testing procedure and types of SUT and evaluated 26 techniques empirically. The authors also mentioned the need to focus on experimental designs and evaluation methods and suggested the use of statistical methods for results evaluation.

In addition to these two SLRs and one maping study, there are three significant surveys in the regression testing domain [Yoo and Harman 2012; Biswas et al. 2011b, Orso and Rothermel 2014]. All these surveys presented detailed and comprehensive disscussions about test case selection, prioritization, and reduction methods. In Yoo and Harman [2012], the authors analyzed 159 studies covering the time period between 1977 and 2007. This survey reported that emerging trends in the field include test data generation, multiobjective regression testing, and test oracles (with cost). The authors also mentioned that experimental designs and empirical evaluations process are not up to the mark, but it is getting the attention of researchers. In Biswas et al. [2011b], the authors compared different regression testing techniques and classified those techniques with respect to application types. This survey highlighted the need to investigate the relationship between cost- and coverage-based effectiveness [Biswas et al. 2011b]. The authors also put emphasis on investigating granularity of coverage, such as statement coverage, branch coverage, condition coverage, and other coverage types. The size of SUTs and test suite size also need to be investigated more precisely. This survey also concluded that the relationship between cost, coverage, and other measurable factors of software testing need to be properly emphasized. Orso and Rothermel [2014] reviewed important studies in the domain of software testing between 2000 and 2014. The survey concluded that there are challenges in domains, such as test oracles, testing modern (big size) application, domain-based testing, testing non-functional properties of software, and probabilistic program analysis. They also mentioned the need to establish repositories for testable artifacts (software systems, test suites, bug fixes reports). This survey also mentioned that more research is required for empirical analysis and design in software testing.

Based on the discussions provided in previous SLRs, mapping study, and surveys, it is obvious that these studies agreed on empirical evaluation for software testing in general and regression testing in particular.

2.1. Regression Test Case Selection

Test case selection methods were first proposed by Fischer [1977] for maintenance of software. Regression test selection attempts to rerun the subset of initial test suites and verify that the changes do not affect the current software version. The main purpose of RTS is to improve the cost of regression testing and maximize possible fault detection ability [Graves et al. 2001]. Test case selection depends on specific and complete conditions called test cases [Sapna and Mohanty 2010]. Test case selection is the process to re-run the most relevant test cases with respect to changes or updates made to SUT [Elbaum et al. 2003]. As shown in Table II, test case selection is used to select a subset of test cases already available for previously executed test cases [Beizer 1995]. Test cases are inputs for the testing process and act as executional conditions with expected outputs [EEE 1990]. The set of two or more test cases, called "test suite," expands with the evolution of software. It is not wise to re-execute all these test cases as it costs a lot in terms of the development, execution, and maintenance of these test suites [Askarunisa et al. 2010]. Thus, an appropriate test case selection technique is useful in regression testing. Before we discuss test case selection and regression testing together, it is better to discuss regression testing first.

Table II. Regression Testing Steps and Their Related Problems

Number	Steps in Regression Testing	Related Problems
1	Select **T′** sub-set of **T**, a set of test cases to re-run on P′	Regression Test Selection/ Prioritization/Reduction
2	Testing **P′** using **T′** to verify the correctness of **P′** with respect to **T**	Test Suite Execution Problem
3	If required, produce **T′′**, a set of new functional or structural test cases for **P′**	Coverage Identification Problem
4	Test **P′** with **T′′** to verify the correctness of **P′** with respect to **T′′**	Test Suite Execution Problem
5	Create **T′′′**, a new set of test cases and test history of **P′**, from **T**, **T′**, and **T′′**	Test Suite Maintenance

Regression testing is formally described by the following: suppose **P** be the current program and **P′** to be the next version of **P**. Let **S** be the current set of specification of **P** and **S**$'$ be the specifications of next versions of **P′**. Similarly, **T** is the test suite for **P,** and individual test cases can be represented by **t**. A regression test case selection technique tries to select test case **T′** subset of **T** in such a way as to fulfill testing specifications **S′** and the changes or modification from **P** to **P′** [Rothermel 1996; Yoo and Harman 2012]. Table II shows the steps taken during regression testing and related problems that need to be solved in each step.

In Table II, **T** represents the test suite, **P** represents the program under test, **P′** represents the modified program, **T′** represents the test suite for **P′**, **T′′** is the test suite for updated versions of **P′**, and **T′′′** is the test suite to verify the results of **T′′**. In the scenario presented in Table II, a program **P** is tested with a test suite **T**. After modifications made in **P**, it is important to verify that the produced **P′** did not adversely impact the functionality of **P**. Thus, there is a need for a modified test suite, which is represented as **T′** and is used to verify the changes or modified parts of **P′**. This issue is known as the test case selection problem, which aims "to select a subset of test cases from the existing test suite to test the modified program." This is also known as test suite execution problem, which establishes the correctness of the modified program.

The next step is to verify the selection process correctness. It is performed by creating another test suite **T′′**, which is used to verify the correctness of testing and test case selection method by several adequacy criteria, such as coverage information, cost of test suite creation, and maintenance- or fault-based adequacy criteria, like fault rate or fault detection capability. The final step of regression testing is to maintain the test suite for future use. This is done by creating **T′′′**, an updated test suite on the basis of adequacy criterion applied in the previous execution.

The main purpose of RTS techniques is to reduce the cost of testing by reducing the executional cost of regression test suites [Leung and White 1991]. The cost of RTS techniques is compared with *retest-all*, which is done "to re-run all the test cases in the previous run." The *retest-all* approach is impractical due to cost and effectiveness of testing process [Mirarab et al. 2012b]. An alternative to the *retest-all* approach is the random test case selection [Chittimalli and Harrold 2009]. The downside of random test case selection is it may fail to identify many regression errors [Biswas et al. 2011b]. RTS techniques attempt to remove the drawbacks of *retest-all* and random test case selection by selecting more relevant test cases to adequacy criterions or testing objectives depending on testing process, such as coverage information [Rosenblum and Weyuker 1997], cost bounds [Graves et al. 2001], or fault-based statistics [Ostrand et al. 2005].

There are many different objectives and possible direct or indirect benefits with regard to regression test case selection. The major goals observed in the literature for regression test case selection are N-release development [Orso et al. 2004], continuous development, continuous integration, and continuous quality enhancement [Lewis 2016]. RTS techniques ultimately contribute directly and in some cases indirectly to overall quality of software product [Lewis 2008], maintenance activities [Wong et al. 1997], reliability of software product [Musa 1993], transition of software system from older systems [Victor 2003], deployment activities of product, software upgrades [Brereton et al. 2007], training of applications as well as staff and version control system. But RTS techniques work with code/requirement information, risk management data, software production, and metrics to evaluate the results of testing process. Due to the multi-facet research domain and problem types, it is necessary to have a framework to assess all these empirical studies with measureable objectives. In the next section, we propose a framework to assess the empirical studies conducted in regression testing in general and regression test case selection in particular.

2.2. Empirical Studies for Regression Test Case Selection

To assess whether RTS techniques can be applied in real-life software projects, there is a need to manage empirical studies that evaluate effectiveness with respect to cost, coverage, or fault detection capability. There are three types of empirical studies, namely case studies, experiments, and surveys [Wohlin et al. 2012]. In software engineering, case study is an investigation that is established on multiple sources of evidences to inquire one precedent for a particular phenomenon [Wohlin et al. 2012; Kitchenham 2004]. Experiment or controlled experiment is an empirical inquiry that manipulates one factor of the studies settings. Surveys have the ability to evaluate a large number of variables by investigating smaller sample of representative variables. Surveys are conducted to knowledge domains, tools, or techniques that are already mature.

After studying the literature on empirical studies [Dyba et al. 2005; Kitchenham et al. 2004; Kitchenham et al. 2002; Johnson 2002], we could not find a model or framework that could help to systematically evaluate the empirical studies. Thus, we felt it is necessary to create an assessment framework to evaluate this kind of study. Our proposed framework will provide a proper quality assessment of empirical studies conducted for RTS techniques as well as help researchers to design their empirical studies for regression testing in the future. The framework has also helped us to assess the quality of the studies selected in this SLR because most RTS techniques are based on heuristic algorithms that add some challenges to measure their effectiveness without a proper framework. The next section presents our proposed framework and explains its components.

3. ASSESSMENT FRAMEWORK FOR EMPIRICAL STUDIES ON REGRESSION TEST CASE SELECTION

The framework below (Figure 1) is composed of three (3) main components: theoretical base of study, scope of study, and evaluation components. Theoretical base of study component covers test problem, test model, SUT, and data set.

This framework provides justification for data collection and selection of empirical studies for this SLR. It does not provide complete operational details and guidelines for the empirical studies, because there is sufficient literature available for software engineering experimental guidelines. For this SLR, we only consider controlled experiments and case studies conducted with the aim to investigate cost, cost-effectiveness, or effectiveness of RTS techniques. This framework presents an empirical study under three granularity levels and refines the process with respect to information collected at each level. This framework also shows the relationship and dependency between these activities. The conceptual framework for RTS regression testing is shown in Figure 1.

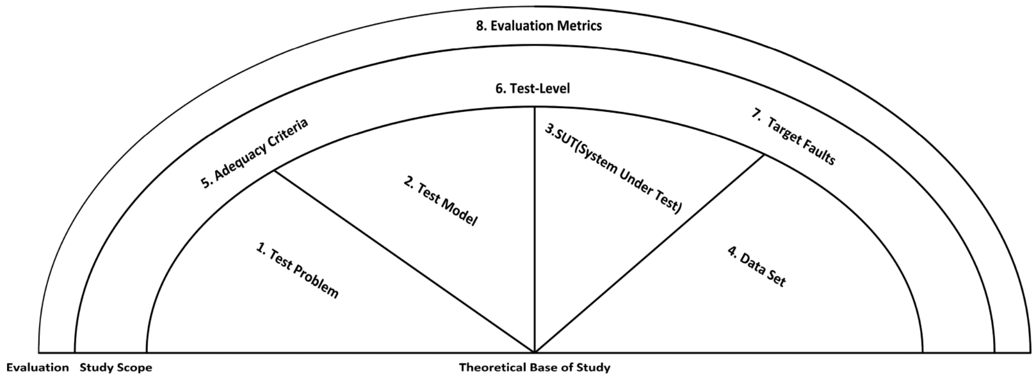


Fig. 1. Conceptual Framework of Regression Test Case Selection Empirical Studies.

3.1. Theoretical Base of the Study

Based on this framework, a case study needs to be assessed based on four core characteristics:

(1) Test Problem,
(2) Test Model,
(3) System Under Test,
(4) Data Set.

The definition for each core characteristic is as follows:

Test problem: This characteristic addresses the theoretical baseline of the empirical study. It covers areas such as formal hypothesis, test purpose, test strategy, and threats to validity [Wohlin et al. 2012]. The problem definition and its success or failure criteria are assessed by the explanation of test problem provided in this framework. The focus of this SLR is on RTS techniques, which focus on effectiveness for which these empirical studies are designed and conducted. Therefore, this characteristic is considered in this SLR.

Test model: This characteristic consists of algorithms or RTS methods and validation criteria. This feature determines the complexity of method evaluated and its verification. The validation criterion listed in this framework can also determine the objective of the empirical study and help to choose other features listed in level two for the design of the empirical study. The testing approach is defined by model being tested. The effectiveness based on cost, coverage, or fault aims the procedure, to systematically exercising the empirical study.

System under test: This characteristic elaborates the application domain for the empirical study. It can also help to distinguish the methods of selection. SUT also influences the choice of dataset for empirical study as well in the sense that either the system under test is object oriented, structured, web service, or mobile application. It affects the testing method and its complexity. Since this is an important parameter for regression testing, it is included in this SLR.

Data set: This is the artifact on which an empirical study is conducted upon. The main feature of any data set is its size. As shown in Table III, a study is considered as small (S) if its size is less than 2,000 lines of code (LOC), medium (M) if it contains from 2,000 to less than 100,000 LOC, and large (L) if it contains more than 100,000 LOC. In most

Table III. Classification Scheme for Size of Empirical Studies

Size Measure	LOC (Lines of Code)	Class Count (CC)	Function Count (FC)
Small (S)	<2000	<100	≤ 1000
Medium (M)	2000 to $<100{,}000$	≥ 100 to <1000	>1000 to $<50{,}000$
Large (L)	$\geq 100{,}000$	≥ 1000	$\geq 50{,}000$

of the primary studies, LOC metric is clearly defined but few studies also considered class count and function count as their artifact size. The studies with less than 100 class count (CC) are considered as small studies, medium studies for 100 to less than 1000 CC, and large studies for those with at least 1000 CC. On the other hand, based on function count (FC), studies containing 1000 function count (FC) or less are considered small studies, medium studies if they contain from 1001 to less than 50,000 FC, and large studies for those with at least 50,000 FC.

The second feature of the data set is size of test suite, which is normally equal to the number of test cases present in a test suite. RTS technique can select test cases from such a test suite. The third feature of the data set is the tool or environment used to conduct the empirical study.

3.2. Scope of the Study

The study scope always narrows down the problem by selecting the test level, target faults, and relevant adequacy criteria. If the scoping is not considered then results of the empirical study cannot be generalized and verified. The following parameters need to be considered for the scoping of empirical studies for RTS techniques:

(1) Test level,
(2) Target faults,
(3) Adequacy criteria.

The definition for each parameter is as follows:

Test level: This parameter concerns the types of testing conducted, for example, unit testing, system testing, smoke testing, and integration testing. There are many reasons why this parameter is considered, among them is because RTS techniques need to be applied during the maintenance and system release phases. It is also evident that test cases like unit testing may also be reused at integration-testing and system-testing phases. Therefore, it is appropriate to consider test levels for RTS application techniques in this SLR.

Target faults: The primary objective of all testing techniques is to uncover faults. There are many types of faults tackled by different RTS techniques. The most common faults focused during regression testing are real faults [Anderson et al. 2014], structural changes [Pasala et al. 2008], and mutational faults [Mirarab et al. 2012a]. Discussion on test case selection mechanism of SLR will be insufficient if faults is not included, thus we included this issue in this SLR.

Adequacy criteria: RTS testing strategy is defined via testing model and adequacy criteria. This arrangement helps to systematically execute the SUT. The common adequacy criteria used in RTS techniques are coverage-based, fault-based, cost-based, and any combination of these three adequacy criteria. These criteria are used as proxy for regression testing and verification purpose as well. The choice of adequacy criteria depends on RTS techniques as well as the target application domain and testing tools.

3.3. EVALUATION (METRICS)

There are a number of existing RTS techniques that vary in terms of their objectives and designs. Due to this, it is difficult to set a common evaluation mechanism for the results of these techniques. Similarly, there are many different metrics used for evaluating these results. The most common metrics in RTS evaluations are the following:

Inclusiveness: This metric "measures the extent to which a technique chooses tests that cause modification program to produce different output than the original program" [Rothermel and Harrold 1996]. **Precision:** This metric "measures the ability of a technique to skip test cases that will not produce different output with respect to original program" [Rothermel and Harrold 1996]. **Recall** is considered as the completeness measure of a RTS technique, and it is measured as percentage of selected failed tests from all failed tests [Chen et al. 2011a]. **Efficiency:** This metric "measures the cost of computation of a technique under study" [Rothermel and Harrold 1996]. ***F-Measure*** is used to measure the fault detection capability and cost reduction of a selection technique [Chen et al. 2011a]. ***F-Measure*** is the combination of *precision* and *recall*, and it is a widely used measure in information science. **FDR:** This measure is used with code-based path analysis RTS techniques to compare each *FDR* path and chooses the most suitable one [Hemmati et al. 2010b]. **Average Percentage Faults Detected (APFD):** This metric assigned the number between 0 and 100 to indicate fault detection. A higher value indicates better fault detection capability, and vice versa. ***APFD*** [Rothermel et al. 2001] is usually used in test case prioritization techniques. **APFDC:** APFD is applied in the cases where test costs and fault severity are not changed. However, the APFDC is being used when test costs and fault severities are fluctuating [Elbaum et al. 2001]. **Average Fault Detection Rate (AFDR):** This metric is derived from APFD, it "enables the overall comparison of two selection techniques in terms of fault detection rate" [Elbaum et al. 2002]. **Multi-Objective Optimization (MOO):** A family of metrics that consist of Hyper Volume (HV) [Deb 2001], Generational Distance (GD) [Coello et al. 2002], Inverted Generational Distance (IGD) [Coello et al. 2002], and Coverage (C) [Deb 2001]. HV computes the size of dominating space, and a higher HV value is considered good in MOO problems. GD and IGD indicate average distance of Pareto Frontier (PF) from actual distance of desired PF value. **Coverage:** This metric presents the number of solutions within no dominated set of comparison algorithms.

The detailed coverage of these metrics is out of the scope of this SLR. The choice of metrics used for the evaluation of an empirical study results is compelled by the nature of RTS solution and SUT. The evaluation of RTS techniques revealed the strengths and weaknesses of the techniques. The type of information collected through these empirical studies depends on the choice of algorithms, tools used for these experiments, SUT, and evaluation processes used to verify these results.

This section has provided the clarification on basic concepts of regression testing, test case selection, test cases, and evaluation metrics. We now present the SLR protocol used to collect the primary studies, analysis procedure, and result documentation process in Section 4.

4. SYSTEMATIC LITERATURE REVIEW (SLR)

In order to carry out the SLR, we adopt three review guidelines steps [Brereton et al. 2007] as shown in Figure 2. This process consists of planning, conducting, and documentation of the results. The first step describes the rationale to carry out the SLR, which is discussed in Section 2.2. The specification of research question is described in Table IV. The primary study selection is presented in Section 4.2, and inclusion and exclusion criterion is given in Section 4.5. The data extraction method is discussed in Section 4.6. Finally, the results are discussed in Section 5.

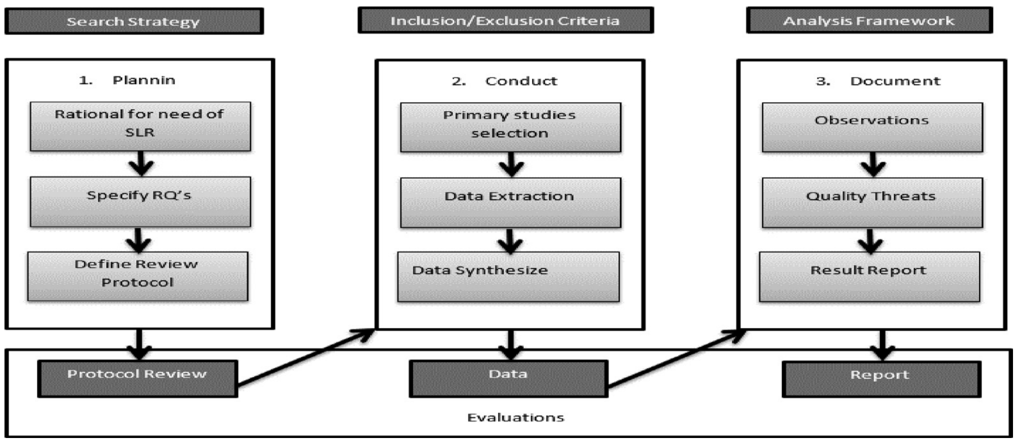


Fig. 2. Overview of Research Methodology for SLR.

Table IV. Research Questions and Their Motivations

Research Question (RQ)	RQ Statement	Motivation
RQ 1	What is the state-of-art research in test case selection techniques in software regression test case selection?	
RQ 1.1	What types of RTS techniques are in practice for controlled experimentations?	These research questions focus to characterize the current domain of test case selection. The reason to form these questions is because the main characteristics of software testing frequently reported in the literature are test level, targeted faults, test model and categorization of test cases with respect to the application domain.
RQ 1.2	At what test levels do these studies conducted?	
RQ 1.3	In what types of application domains/environments do these studies conducted? What types of faults do they focused on?	
RQ 2	How were the empirical studies designed and conducted?	
RQ 2.1	How do the empirical studies designed for test case selection techniques?	The empirical studies which are objective critical and have proper design are easy to assess and replicate reliable results with reproducibility. The sub-questions help to assess the empirical studies in terms of their design.
RQ 2.2	What types of data sets/objects/artifacts were used for the empirical study?	
RQ 3	What empirical evidence is available for the cost, coverage and fault based effectiveness of RTS techniques?	
RQ 3.1	What empirical evidence is available for the cost or cost effectiveness?	These questions help to incorporate the outcomes shared by the previous studies. These research questions can also help to check the level of empirical evidence obtained after this study has been conducted. We also attempt to find answers to the sub-questions to synthesize the main research question.
RQ 3.2	Is it possible to collect some empirical evidence on the effectiveness of RTS techniques?	
RQ 3.3	What types of evaluation methods were used to verify the studies results?	

4.1. Research Questions

The research questions are framed after discussions have been made with testing experts and experienced authors. The objective of these research questions is to find the most effective and relevant regression test case selection techniques, focusing on cost, coverage, and fault detection. In the same time, we want to find out techniques used for the empirical evaluations. With the above-mentioned objectives, the authors also tried to focus on measurable aspects and techniques that had sufficient evidence with reasonable size of projects. The questions are shown in Table IV.

4.2. Strategy for Study Selection

This is an important step for any SLR that ensures completeness of the selection of primary studies. The quality of results is usually based on primary studies selected for the SLR. Selection for our study was based on three steps:

(1) Selection of source repositories,
(2) Identification of search keywords,
(3) Inclusion or exclusion criteria for studies based on the research questions.

4.3 Selection of Source Repositories

We initiated this process by entering search strings on source databases. These databases will return a set of studies. The retrieved studies were then mapped to the inclusion and exclusion criteria. Selection of appropriate databases and search strings are quite important for SLR quality because they directly influence the completeness of the results of the study. We used the following popular repositories:

(1) IEEE Xplore,
(2) ACM Digital Library,
(3) ScienceDirect (including Elsevier Science).

The rationale behind choosing these repositories was that IEEE Xplore and ACM Digital Library covered almost all important conferences, while ScienceDirect covers almost all important journals in the domain of software engineering, especially software testing.

In this section, we present the source repositories, and in the next section, we present the search keywords used to search these repositories.

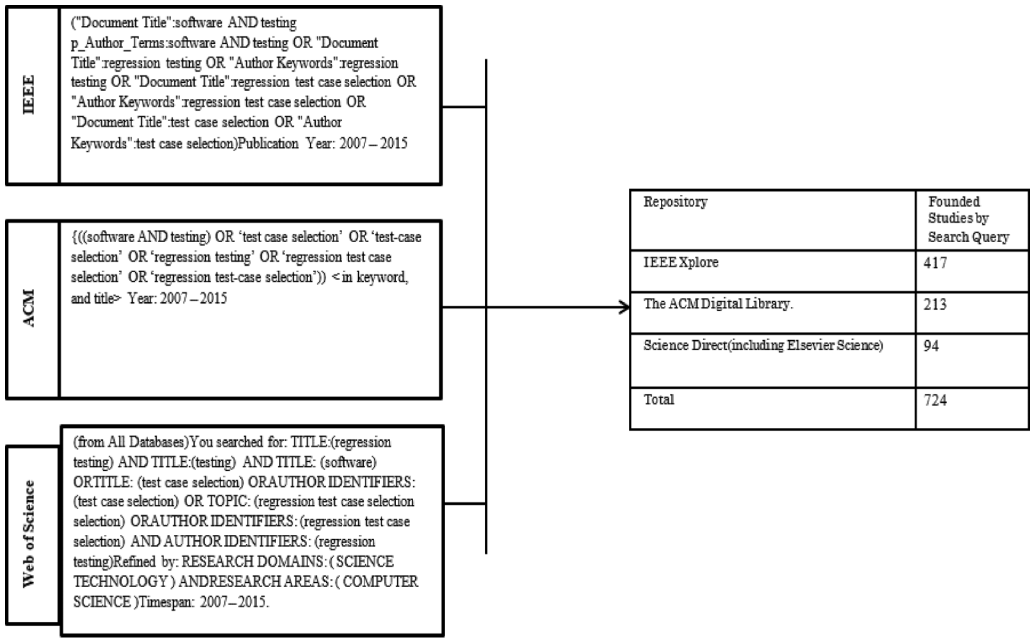
4.4. Identification of Search Keywords

The systematic method used to formulate the search keywords cosists of the following steps:

(1) Identify major keywords based on research questions,
(2) Identify alternative words and synonyms for the major keywords,
(3) Frame a search string by joining these keywords with Boolean AND operator, for alternative words use the Boolean OR operator.

Since our main focus was to investigate empirical studies in RTS, the following major keywords were used for the purpose of finding relevant studies in the field of software testing and test case selection. We intentionally avoided empirical study as a keyword due to the fact that most studies do not use this keyword. Initially, we collect the keywords from available research studies in the domain of regression test case selection, and then we use these keywords to start the process of designing search queries. The second source of collecting initial search strings are previously published SLRs in the domain of software testing.

As shown in Figure 3, we keyed in a number of search strings on different repositories. In order to retrieve maximum possible relevant studies, we used the following terms: “software testing,” “regression testing,” “test case selection,” and “regression test case selection.” It should be noted that if we use keyword “testing” alone, then it returns too many irrelevant studies. Due to this observation, we used AND operator between “software” and “testing” keywords. This expression finds maximum relevant studies on the topic under investigation. To obtain the most relevant search results, authors also



IEEE

("Document Title":software AND testing p_Author_Terms:software AND testing OR "Document Title":regression testing OR "Author Keywords":regression testing OR "Document Title":regression test case selection OR "Author Keywords":regression test case selection OR "Document Title":test case selection OR "Author Keywords":test case selection)Publication Year: 2007 – 2015

ACM

{((software AND testing) OR 'test case selection' OR 'test-case selection' OR 'regression testing' OR 'regression test case selection' OR 'regression test-case selection')) <in keyword, and title> Year: 2007 – 2015

Web of Science

(from All Databases)You searched for: TITLE:(regression testing) AND TITLE:(testing) AND TITLE: (software) ORTITLE: (test case selection) ORAUTHOR IDENTIFIERS: (test case selection) OR TOPIC: (regression test case selection selection) ORAUTHOR IDENTIFIERS: (regression test case selection) AND AUTHOR IDENTIFIERS: (regression testing)Refined by: RESEARCH DOMAINS: (SCIENCE TECHNOLOGY) ANDRESEARCH AREAS: (COMPUTER SCIENCE)Timespan: 2007–2015.

Repository	Founded Studies by Search Query
IEEE Xplore	417
The ACM Digital Library.	213
Science Direct(including Elsevier Science)	94
Total	724

Fig. 3. Search Strings for Selecting Studies from IEE, ACM, and ScienceDirect.

switched the search string with OR to cover document titles, author keywords, and time span between 2007 and 2015.

When executing these search queries on the selected databases, it was also found that ACM Digital Library also indexed selected ScienceDirect and IEEE Xplore sources, and ScienceDirect also indexed IEEE Xplore listed journals. These redundant results were removed from the search output. Furthermore, these selected databases employ quite distinct search criteria, so where possible, we try to refine our selection by narrowing our selection of sources, especially in the ScienceDirect database.

The reason of doing so is to get the most relevant research studies for analysis; otherwise, the number of retrieved studies was so high it made it difficult to perform further analysis. In the next section, we elaborate the inclusion and exclusion criteria for primary study selection from studies obtained from the selected source repositories.

4.5. Study Selection Based on Inclusion and Exclusion Criteria

RTS have been used with many different features and many software testing types with respect to different testing objectives and application types. At the same time, it is also a superset of other test suite optimization techniques, such as test suite prioritization, test suite reduction, and test suite augmentation. There are also many features on which these RTS methods and techniques are proposed and evaluated, such as size of SUT, size of test suite, code-based changes, specification-based changes, efficiency, and application-failure-based validation. As mentioned above, the focus of this SLR is to investigate the effectiveness of RTS techniques, including cost, coverage, or fault detection ability. Thus, it is necessary to define the inclusion and exclusion criteria that help to select the relevant primary studies. In this section, we explain our inclusion and exclusion criteria and reasons behind the selection:

(1) We executed our search queries on selected repositories as mentioned before. We found 724 research studies between 2007 and 2015 (after removing duplicate

Table V. Distributions of Studies After Applying Inclusion/Exclusion Criteria

Repository	Studies Found	First Level	Second Level
IEEE Xplore	417	144	26
ACM Digital Library	213	39	08
ScienceDirect (including Elsevier Science)	94	32	13
Total	724	215	47

results). We followed two stage selection processes as shown in Table V. In the first stage, studies were selected based on their titles and abstracts. If a study title or abstract did not include RTS technique focusing on effectiveness (cost, coverage, or fault detection ability), then that study is filtered out/excluded at this stage.

(2) The title or abstract that did not include RTS method/technique were excluded.
(3) The title or abstract that did not discuss the automation of RTS technique with respect to cost- or coverage-based effectiveness or fault detection was excluded.

If there exists any confusion in the first phase of selection, the study will be brought forward to the second phase for further refinement. After first phase, we were left with 215 studies. At the second level of inclusion/exclusion, the studies were divided into three sets according to their research questions. We studied their contents in detail, one by one. We exclude several studies based on the following exclusion criteria:

(1) Posters, technical reports, PhD dissertations, and studies with less than 5 pages were excluded. The primary goal of this study is to build a synthesis on peer-reviewed studies with sufficient technical details.
(2) Studies that did not focus on RTS effectiveness (cost, coverage, fault detection) methods/techniques were excluded.
(3) Studies that did not report any empirical evidence on the focus areas (cost, coverage, fault detection) were excluded.

In some cases, during the second level, we were not able to decide whether to include several studies in this SLR. To make this important decision, we discussed it with other researchers and decisions were made based on consensus. The one important point about selection process is we cannot exclude studies on the basis of the SUT. The reason behind this decision was by considering this exclusion parameter, the domains were narrow and only few studies would have been selected. After phase two, we were left with 47 studies. Out of these 47 studies, three articles on empirical study were extended versions of a conference article published in some journals. In this case, we consider these three articles as one.

In the next section, we will present data extraction method from primary studies collected from the inclusion/exclusion criteria.

4.6. Data Extraction Method

After the relevant studies were selected, data extraction forms were designed to collect data from these studies. Data needed were collected in two phases and into two separate sets of data sheets. The first sheet contains standard information [Kitchenham 2004], which includes study title, authors name, brief summary, and comments by researchers performing the studies. The second sheet consists of information directly extracted from the studies. The information collected in the first set was used for the inclusion/exclusion criteria, while the information collected in the second set was used to answer the research questions proposed. The mapping of data collection with their respective RQs is shown in Table VI.

In the next section, we will present results of the SLR from the primary studies data collected.

Table VI. Data Collection for Research Questions Framed

Research Questions (RQs)		Type of Data Extracted
RQ 1	RQ 1.1	Types of RTS techniques, solution types, or algorithms used to conduct RTS techniques
	RQ 1.2	Test level of the empirical studies conducted
	RQ 1.3	Targeted faults, test type, and application domain or process model
RQ 2	RQ 2.1	Formal hypothesis, test purpose, test strategy, threats to validity
	RQ 2.2	Size of dataset, size of test suite, tool/environment for experimental setup, number of test suite runs, data collection method
RQ 3	RQ 3.1	Evidence type or measure for cost or cost effectiveness of RTS technique
	RQ 3.2	Evidence type or measure for effectiveness of RTS technique
	RQ 3.3	Metric or comparison baseline used to validate empirical study results

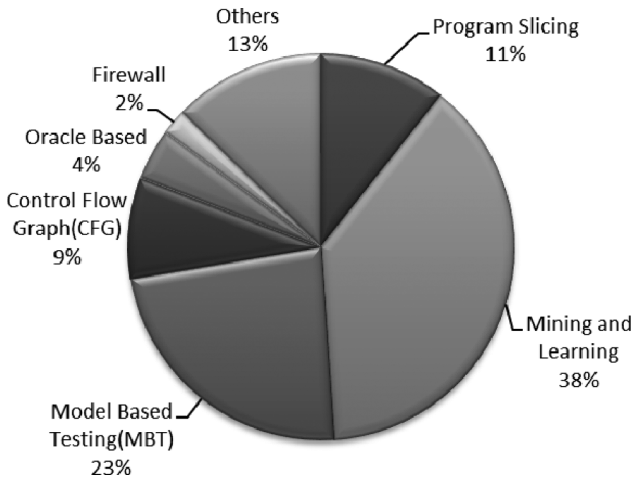


Fig. 4. RTS Techniques Used in the Studies.

5. RESULTS

This section outlines the results with respect to the research questions.

5.1. RQ 1: What is the State-of-the-Art of Research in Test Case Selection Techniques in Software Testing?

The purpose of this research question is to assess state-of-the-art research in regression test case selection techniques. In order to answer this question, data were collected from primary studies with the help of SLR protocol as shown in Figure 2, and the studies were further assessed with the help of framework as shown in Figure 1.

5.1.1. RQ 1.1: What type of RTS Techniques Were in Practice for Controlled Experimentations? The first aspect of this RQ was on RTS techniques categories based on their commonalities in selection procedure, input type, and output type. We grouped these RTS methods in five categories listed below. These categories are reported in recent surveys and they have lots of common characteristics [Biswas et al. 2011a]. The second reason to group these techniques into these categories is that each study uses different selection procedure with similar objectives and constraints. The percentage of the reported techniques is shown in Figure 4. The techniques used in the studies are as follows:

(1) Mining and Learning,
(2) Model Based Testing,

(3) Program Slicing,
(4) Control Flow Graph (CFG),
(5) Firewall,
(6) Other techniques.

The results showed that mining and learning-based RTS techniques are most utilized method between 2007 and 2015. They were used in 18 of 43 (38%) studies. Genetic Algorithm (GA) is used in 5 of 17 studies [S17, S20, S22, S31, S33]. Several observations are noted for the GA utilization. First, there are many publications on GA application and it is used for different problems. Second, empirical data is easily available for GA experimental setup. This eases researcher in executing and compiling results using GA algorithm. Fuzzy rule-based reasoning is used in 5 studies [S6, S15, S34, S35, S45]. Fuzzy logic has appropriate mathematical reasoning under certain conditions. It also removes unwanted precision from analysis [S35]. Fuzzy logic can also accommodate the form of rule-based reasoning, valuable project experience gained by testers during project execution can be transformed into fuzzy rules, and remove drawbacks of linear modeling [S35]. Practical swarm optimization (PSO) was reported in 4 studies [S19, S23, S24, S37]. It was reported for multi-objective selection and optimization criteria for code coverage, functional requirement with cost, and effectiveness constraints. It is also reported [S19] that PSO gained more attention compared to GA and other search-based algorithms on performing RTS methods.

The second-largest portion of reported technique in our primary study is MBT in RTS techniques (26%) [S2, S12, S13, S16, S25, S27, S28, S29, S30, S42, S43]. These UML-based RTS techniques are used in collaboration diagrams, sequence diagrams, state charts, use-case diagrams, and class relationship diagrams. Most of the time, these techniques use control flow, function calls, behavior changes, component changes, data dependency, and message sequences or message changes for their selection criterion. Most of these studies represent artifact designs and do little to the code or application level testing. It is also observed that MBT methods are used with GA techniques [S20, S28, S30] for some common objectives like similarity-based measures and test case diversity measures.

Program Slicing RTS techniques are used in 11% [S1, S3, S39, S40, S47] of the primary studies and they are the pioneer RTS techniques [S1]. Slicing techniques mostly omit test cases that do not produce new, relevant outputs. They are called nonmodification revealing test cases [S40].

From the analysis, these techniques are applied to small- or medium-sized data sets, which are mostly procedural or modular applications. They are also difficult to be scaled from one environment to another environment [S3].

CFG-based RTS techniques have a share of 9% [S7, S10, S38] of the primary studies. These techniques are based on flow control of programs or workflow of program behaviors and functions. These techniques are also applied to small datasets and only cover changes within the execution flow and do nothing to those outside of the current flow [S7]. They also worked with coverage information [S7, S38] and updated coverage information [S10] with program path analysis.

Oracle-based RTS techniques are used in 4% [S8, S14] of the studies and mostly focus on test case profiles or execution profiles of test cases. These techniques try to predict the fault rate or effectiveness of the test suites for future use. The creation and maintenance of such profiles are considered as overhead and consume extra resources in oracle-based RTS techniques. It is also observed that an emerging domain other than oracle-based RTS, but with the same goal of oracle-based RTS using selection procedures of basic selection techniques. These techniques are named in test case

Table VII. Test Case Profiling and Selection with Historical Data Analysis Techniques

Technique	Description	Studies
Mutation Score	Calculate mutation score for each test case and select the best score	S22, S26, S45
Historical Data Analysis	Utilize the coverage information and fault revealing power or execution time constraint with historical data analysis to select test cases for current programs	S30, S32
Test Case Profiling Based on Numerical Data Analysis	Maintain a profile for test cases on coverage information, fault rate or fault revealing test cases, and defect discovery time	S6, S10, S15, S31, S40
Test Case Ranking	A statistical model based on earlier testing results (fault potency and fault rates). The model can be used to determine the next sets of test cases	S2

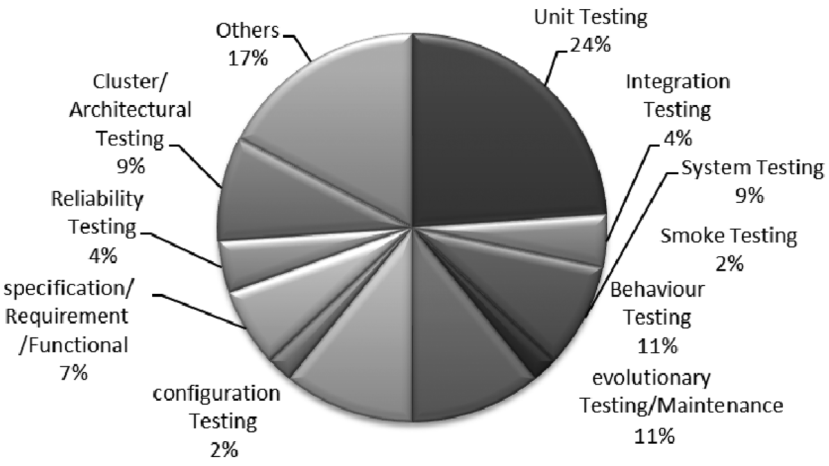


Fig. 5. RTS Testing Level.

profiling and selection with historical data analysis, test case ranking, or test case selection on the basis of mutation score as shown in Table VII. The commonality between all these techniques is the analysis of previous test data collected through previous testing cycles.

Firewall technique [S9] consists of 2% of the primary studies. Other individual RTS techniques, which are mostly based on particular experimental setups, make up 12% of our primary studies.

5.1.2. RQ 1.2: At What Test Levels Were These Studies Conducted? The second aspect is the testing level of these primary studies as shown in Figure 5. The analysis shows that Unit testing recorded the biggest percentage, which is 24% of all studies, followed by Behavioral testing (11%), Evolutionary or Maintenance testing (11%), System testing and Cluster/Architectural testing (9%), Specification/Requirement/Functional testing 7%, Reliability testing and Integration testing (5%), and, lastly, 17% of the studies that did not mention any particular testing level.

It is observed that the type of testing parameters used depend on the testing framework, for example, JUnit is used by Java. Thus, the choice of testing environment leads to the choice of unit testing for most studies. The second possible reason for the choice of testing level was the dataset under test. It is observed that most datasets are publicly available and can be used for unit tests.

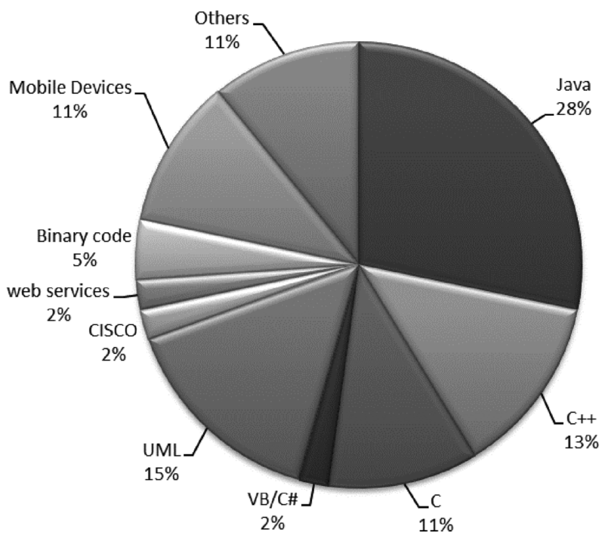


Fig. 6. RTS System Under Test.

5.1.3. RQ 1.3: What Types of Applications/Environments and Fault Types Were Conducted in These Studies? The third aspect is about SUT/environment where these empirical studies were performed. We observed that Java is the most frequent language used in these studies (28%), followed by UML-based artifacts (15%), C++ (13%), mobile devices and C (11%), binary codes (5%), and, finally, VB/C#, web services, and CISCO devices (2%) as shown in Figure 6. In this case, the factor of dataset availability and testing artifacts may lead to the reason why Java is chosen. This is because most open source applications are written in Java. Apache and SIR are two well-known test artifact repositories having information of testing information with fault reports. Majority of existing datasets are based on Java and JUnit framework.

The fourth aspect was the fault type that these primary studies were using. We observed that 55% of the studies did not use any particular fault type but used some structural coverage criteria to fulfill RTS procedures. The structural coverage criteria used in the studies were code coverage, condition coverage, modified condition coverage, function coverage, and constraint coverage. The second largest proportion of fault types used in these primary studies were real faults/mutational faults (33%), followed by code changes (16%) as shown in Figure 7.

5.2. RQ2: How Were the Empirical Studies Designed and Conducted?

The type and design of datasets used for empirical evaluation are important. A technique's level of effectiveness does not guarantee its suitability to complete testing objectives. Thus, in order to conduct an effective experiment, there is a need to follow experimental guidelines for experiment design. In this research question, we investigate the reported studies' designs, reproducibility, and scale.

5.2.1. RQ 2.1: How Are Empirical Studies Designed for Test Case Selection Techniques? In order to answer this question, the primary studies were classified into two classes. We used the following definitions for the experiments: "A study in which an intervention is deliberately introduced to observe its effects" [Shadish et al. 2002]. Case Study: "An empirical inquiry that investigates a contemporary phenomenon within its real life context, especially when the boundaries between phenomena and context are not clearly evident" [Yin 2003].

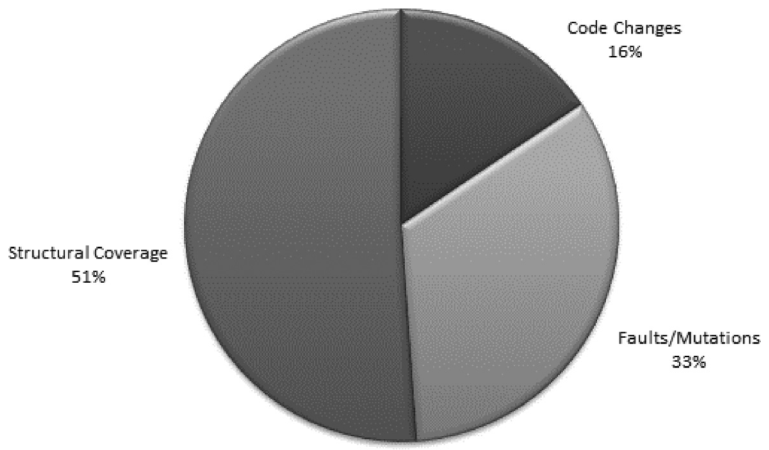


Fig. 7. Fault Type Proportion.

Table VIII. Experiment/Case Study Setup Details

ID	Question	Yes	Partially	No
1	Are the research questions clearly stated?	18 (41%)	25 (58.1%)	0 (0%)
2	Were threats to validity clearly stated?	14 (32.5%)	12 (27.9%)	17 (39.5%)
3	Is the study repeatable (public datasets)?	16 (37.2%)	10 (23.2%)	17 (39.5%)

The analysis shows that 65% of primary studies claimed that they were evaluated as case studies, as shown in Table IX. The remaining 35% studies are claimed as experiments. It is also observed that 18 of the 47 studies (38%) clearly mentioned their research questions and 25 of the studies (53%) partially mentioned their research questions. Only 14 studies (29%) mentioned their internal and external threats, another 12 studies (25%) partially mentioned these threats, while the rest (17 studies or 36%) did not mention this aspect. There were 16 studies (34%) that used public datasets for case study/experimentation, 12 studies (25%) used open source or partially available datasets, while the rest (17 studies or 36%) used private datasets, making them not reproducible. The overall observation was that during this analysis, empirical design was not considered as its due share in proper software-testing experiments. The second observation was that research questions framed for investigating the problem were too general and often did not return with sufficient information. The threats to validity or study limitations in such studies were also ignored in a majority of the studies.

5.2.2. RQ 2.2: What Types of Datasets/Objects/Artifacts Were Used for the Empirical Study? Size of the case study/experiment is an important factor to assess their validity. In order to answer this question, we classify the studies into small, medium, large (S, M, and L) studies. The size was determined based on the criterion presented in Table III. The details on the studies' artifacts, types, and sizes are presented in Table IX. The results show that 12 studies were made on large-size datasets, 12 studies on medium-size datasets, and 16 studies on small-size datasets. There were 7 studies that did not report their size of datasets. It is also observed that 13 studies used datasets taken from SIR [Infrastructure 2016], 7 were using industrial datasets, and 13 studies did not mention their source of dataset. It is observed that the use of publically available datasets increases since 2007. This trend is due to the existence of dataset repositories.

5.3. RQ 3: What Empirical Evidence Is Available for the Cost-, Coverage-, and Fault-Based Effectiveness Measure of RTS Techniques? It is difficult to measure the costs and effectiveness

Table IX. Detail of Primary Studies, Artifacts, and Study Type and Size

#	Ref	Artifact	Type	Size
S1	[Tao et al. 2010a]	Details not reported	EXP	S
S2	[Tsai et al. 2009]	60 web services, Own unspecified	EXP	M
S3	[Tao et al. 2010b]	SIR (Siena, jtopas)	CS	S
S4	[El-Hamid et al. 2010]	Calculator & sorting, Own unspecified	CS	S
S5	[Huang et al. 2011]	Configuration files	EXP	S
S6	[Xu et al. 2013]	Real life telecom project	CS	L
S7	[Xu and Rountev 2007]	Seven programs: Bean, tracing, telecom, quicksort, nullcheck, dcm, lod	Exp	S
S8	[Yu et al. 2013]	SIR (Make, grep), ABB (5 programs)	CS	M
S9	[Zheng et al. 2007]	ABB application	CS	L
S10	[Chittimalli and Harrold 2009]	Java apps, Own unspecified	CS/Exp	L
S11	[Rachatasumrit and Kim 2012]	SIR (Jmeter, XMLSecurity, ANT)	CS	M
S12	[Pasala et al. 2008]	inARTS tool, 7 components of Windows XP based app & NunitForms (open source)	CS	M
S13	[Fourneret et al. 2014]	Details not reported	CS	S
S14	[Nanda et al. 2011]	Prototype tool TREND and Consultant Assistant programs, Details not reported	CS	M
S15	[Chen et al. 2011a]	Schedule (program), SIR (Flex, space), gcov (a GNU tool)	CS	M
S16	[Iqbal et al. 2010]	Student enrolment system	CS	S
S17	[Mirarab et al. 2012a]	SIR (Ant, Nano, Galileo, Jmeter)	CS	M
S18	[Pang et al. 2013]	SIR (Nanoxml, jtopas, Jmeter, xml-security, .ant)	CS	S
S19	[De Souza et al. 2011]	2 applications, Details not reported	Exp	M
S20	[Hemmati and Briand 2010]	Safety monitoring component	CS	M
S21	[Cibulski and Yehudai 2011]	Open source Apache based (Log4j, commons Math)	CS	M
S22	[Assis Lobo de Oliveira et al. 2013]	5 benchmarks (polo)	CS	M
S23	[De Souza et al. 2014a]	SIR (space)	CS	S
S24	[De Souza et al. 2014b]	Details not reported	EXP	
S25	[Hemmati et al. 2011]	Safety monitoring component, video-conference system, TRUST tool for testing	EXP	
S26	[Delamaro and Offutt 2014]	30 programs Siemens, text books	EXP	S
S27	[Anderson et al. 2014]	Microsoft Dynamics AX	CS	L
S28	[Hemmati et al. 2010b]	Safety monitoring component	EXP	M
S29	[Huang et al. 2009]	General Java application	EXP	S
S30	[Hemmati et al. 2010a]	Industrial Project, Details not reported	CS	L
S31	[Yoo and Harman 2007]	SIR (printtokens, printtokens2, schedule, scheduler, space)	EXP	S
S32	[Yu et al. 2010]	Financial management system	CS	L
S33	[Panichella et al. 2015]	SIR (11programs), Siemens Suite (printtokens, printtokens2, scheduler, scheduler2)	CS	L
S34	[Kumar et al. 2013]	SIR (Print_tokens, Print_tokens2)	CS	S
S35	[Xu et al. 2014]	Telecommunications system	EXP	L
S36	[Rogstad et al. 2013]	SOFIE (tax accounting system)	EXP	L
S37	[De Souza et al. 2013]	Mobile app, Own unspecified	CS	
S38	[Li et al. 2012]	Custom service, Details not reported	EXP	
S39	[Lin et al. 2012]	Details not reported	EXP	
S40	[Chen et al. 2011b]	SIR (space)	CS	S
S41	[Singh et al. 2010]	Calendar, triangle, time-date, Kmap generation, tax, calculation	CS	S
S42	[Mansour et al. 2011]	Custom application, Details not reported	CS	
S43	[Cartaxo et al. 2011]	Details not reported	CS	
S44	[Gligoric et al. 2015]	32 open source projects from Apache, GoogleCode and GitHub	Exp	L
S45	[Shi et al. 2015]	17 open source projects from Apache, GitHub	Exp	L
S46	[Kumar et al. 2015]	SIR (Print_tokens, Print_tokens2)	Exp	S
S47	[Nardo et al. 2015]	NoiseGenSys, Details not reported	CS	L

Table X. Distribution of Cost Measures Used in the Studies

Number	Method	Description	Studies
1	Measuring cost of analysis	This analysis is used to detect changes in code by the compiler (in seconds). The compile time is considered as the cost for the test case selection.	S7
2	Test suite size	This method is performed by removing redundant test cases. It can be based on cost, coverage or fault based effectiveness.	S2, S3, S12, S15, S16, S20, S25, S28, S40
3.	Test suite execution time	By reducing overall test suite execution time by achieving some goal like fault rate (mutation score) or coverage criterion.	S8, S10, S23, S24, S26, S28, S29, S30, S31, S32, S33, S34, S39
4	Defect discovery time	The time to discover defects by the test suite.	S6
5	Bounded cost of execution time	The test suite is executed with limited resources like execution time or mutation score with adequacy criterion.	S21
6	Executional cost of each test case	This method measures executional cost of each test case and make a selection of test cases with reduced cost with adequacy criterion.	S37
7	Test case selection time	Time taken to select the test suite is considered as the cost.	S39

of RTS techniques because they are context-dependent. There are many direct and indirect measures available, especially in measuring a technique's effectiveness. The context-dependency of cost measures and many different cost measures (time, size, effort) also put high hurdles in comparing these costs. Furthermore, these cost measures are normally compared with *retest-all* or random test selection costs, which provides a weak comparison. Due to this, it is almost impractical to compare between different types of studies. The datasets used by these studies are small, so it is not possible to use these results for large-scale projects.

5.3.1. RQ 3.1: What Empirical Evidence Is Collected for the Cost or Cost-Effectiveness or Effectiveness? Assessing cost, coverage, or fault detection ability of RTS methods are the primary objectives of this study. The empirical studies that are focusing on measuring cost, effectiveness, or cost-effectiveness are selected for analysis. As mentioned before, this SLR focuses on measuring cost, coverage, or fault-detection ability of RTS methods. Thus, the measurement of cost, effectiveness, or cost-effectiveness in a valid manner is important. Costs are measured due to the following reasons:

(1) To compare several techniques and to decide which technique is more effective than the other,
(2) To assess whether a technique is practical within a given time constraint.

Seven types of cost measures were used in the 47 studies investigated as shown in Table X. It was observed that 30 studies (63%) measure the cost of RTS methods using one way or another, 16 studies (34%) do not measure cost in any way, while the most dominant trend is to use this measure as a comparison with other RTS techniques. The rest of them (19 studies or 40%) [S6, S10, S12, S15, S19, S20, S21, S22, S24, S25, S26, S28, S29, S30, S31, S34, S37, S39, S40] consider cost as their effectiveness measure.

We observed that test suite execution time is the most popular method used to measure the cost of RTS techniques. This method is reported in 17 of the 47 empirical studies. But the usage of this method is context-dependent and varies between empirical studies. The second-most reported cost measure is test suite size, which is reported in 9 studies. This method is considered as an effectiveness criteria in terms of the reduced number of test cases in the resulting test suite. Study [S28] can use two cost measurement methods, either test suite size or test suite execution time. Meanwhile, study [S39] uses test cases selection time or test suite execution time to measure its overall cost. Relative terms of efficiency and effort are also used to measure cost.

5.3.2. RQ 3.2: Is It Possible to Collect Some Empirical Evidence on the Effectiveness of RTS Techniques? A test suite can be accurately assessed based on its ability to discover faults or fulfill maximum coverage. Measurement of effectiveness varies across different empirical studies being analyzed. Effectiveness with regard to cost, coverage, or fault-detection is the main goal of this SLR. Thus we only select RTS studies that measure effectiveness based on those criteria (cost, coverage, or fault detection). In our case, effectiveness is measured based on two categories, but the measures are very much context-dependent and consider many different parameters that are mixed with each other. The two general categories are as follows:

(1) Coverage-Based Measures of Effectiveness: In this type of measure, some adequacy criterion based on coverage is used to determine the effectiveness of RTS techniques. This measure also varies in terms of its use coupled with many other cost and fault-based measures. Subcategories of this measure include statement coverage, branch coverage, method or function coverage, condition coverage, requirement coverage, and test case coverage, as shown in Table XI.
(2) Fault-Based Measures of Effectiveness: This measure type determines the results based on faults detected by RTS methods using many variants, such as fault detection capability, fault relieving capability, failure rate, regression failures, and many more. The most basic and frequently used measures are shown in Table XII. These measures are also used with historical data analysis to improve RTS techniques' capability for future use.

Overall, 27 of 47 empirical studies (57%) measure effectiveness in terms of coverage using one way or another. From this number, 19 studies [S6, S10, S12, S15, S19, S20, S21, S22, S24, S25, S26, S28, S29, S30, S31, S34, S37, S39, S40] measure effectiveness based on cost. Effectiveness is measured for two reasons, either to select the test cases using effectiveness criterion or to compare this measure with other techniques or previous version of the same dataset.

The most frequently used adequacy criteria observed in the studies are coverage criteria, which correlate with fault-based adequacy criteria or cost measures that are used in 17 studies. It is observed that single criteria is not used at all, and a combination of cost, coverage, or fault-detection ability are used as effectiveness measures in all studies. In coverage-based adequacy type, statement coverage is the most frequently used criteria. The details are shown in Table XI.

The main reasons why statement coverage is used as adequacy criteria is because it is simple, measurable, and devises test cases on the internal logic and structure of the application under test. Statement coverage describes the degree to which a software is being tested. It also provides information about test cases and statements related to them. Other types of coverage based adequacy criteria include branch coverage, method/function coverage, class coverage, change point coverage, and requirement coverage.

There are 8 of the 47 studies that used fault-based adequacy criteria without coverage criteria, as shown in Table XII. The other fault-based criteria used with coverage-based adequacy criteria are shown in Table XI. It is observed that multi-objective criteria-based adequacy measures are common in RTS research domain with mining and learning family of RTS methods. The multi-objective goals are described with fitness function and (dis)similarity measure functions in machine-learning RTS techniques. The

Table XI. Distribution of Coverage Adequacy Criteria in the Studies

Number	Adequacy Measure	Description	Study
1	Statement coverage	Statement coverage used for dimensionality reduction with an execution profile of test cases	S18, S40, S45
2	Statement coverage with fault history	A multi-objective selection criterion developed using statement coverage with fault history and executional cost	S31
3	Statement coverage, branch coverage with fault detection capability	Fitness function based on statement coverage, branch coverage, fault detection capability and executional cost	S34, S46, S47
4	Statement coverage, branch coverage with modified statement coverage	Statement coverage, branch coverage with average rate of fault detection used as the effectiveness measure	S41
5	Updated coverage (statement, branch)	Updated coverage data used as subsequent selection task	S10
6	Additional statement coverage with fault rate	Additional statement coverage with fault detection rate (FDR) as the effectiveness measure and selection task	S25
7	Method/Functional coverage	Method or functional coverage with change impact traceability used for selection purpose	S14, S39
8	Minimized sum of coverage (class, methods, statement)	Multi-objective selection based on minimum sum of coverage data used by the voting mechanism for selection purpose	S17
9	Requirement coverage	Multi-objective with maximizing requirement coverage with minimum cost of execution time of test cases	S19, S37
10	Change point coverage	Change-points coverage with minimum cost measure used to verify program subset	S29
11	Test case coverage with fault detection rate	Test case coverage with fault detection rate used as similarity measure between test cases for selection purpose	S30
12	Path coverage	Path coverage used for identifying changes select test cases	S38
13	Transition-based coverage with fault-based coverage	Similarity measured with transition based and fault based coverage that reduce test suite size by discarding similar coverage test cases	S43

history-based profile of test case execution, fault data repositories, and change identification techniques with coverage and updated coverage information were used for selection or verification of selection techniques results. The other reason to use fault detection ability as effectiveness measures is the ability of testing tools like JUnit and mutation generators. These tools simply measure and display the faults or errors with failed test cases. One thing to note is the degree of measurement varies with tools choice.

5.3.3. RQ 3.3: What Types of Evaluation Methods Were Used to Verify the Studies Results? RTS techniques differ in terms of their goals and selection procedures as mentioned in Figure 1. If we need to choose an RTS technique for practical application, a mechanism is needed to compare and evaluate the results of each technique. It is very difficult to compare the difference in terms of each technique's goals and philosophy. A study [Rothermel and Harrold 1996] identified different RTS categories, which include

Table XII. Distribution of Fault-Based Adequacy Criterion in the Studies

Number	Measure	Description	Study
1	Fault Revealing Capability	Defect discovery capability measured and compared with *retest-all* for effectiveness indicator	S5
2	Defect Discovery Time	Test case execution profile with defect discovery time used as effectiveness measure	S6
3	Fault Detection Capability	The function call profile with the fault detection capability with the goal to reduce cost is used an effective measure	S15
4	Failure Frequency Rate	Most frequent failures with relationship to test cases are used as effectively measure	S26, S27, S35, S36
5	Fault Detection Rate	Fault detection rate with cost of analysis used as effectiveness measure	S28

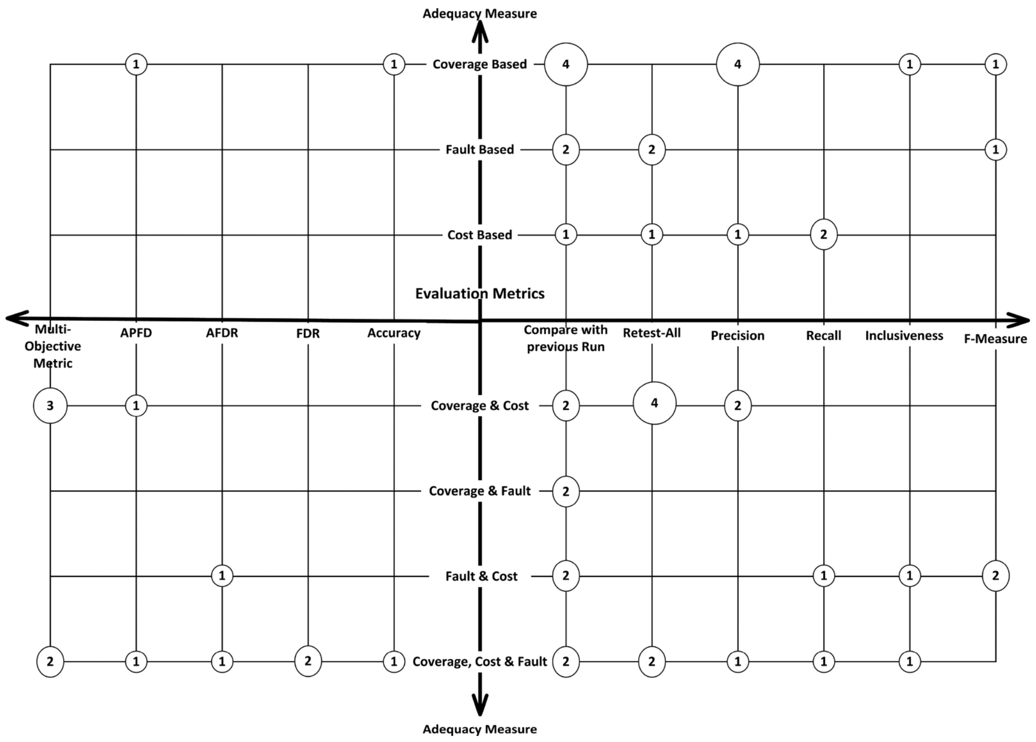


Fig. 8. Adequacy Measure and Evaluation Method Perception Map.

inclusiveness, precision, efficiency, and generality. There are many other metrics used to compare the results of RTS techniques studies. Possible classes to be used in the comparison is given in in Section 3.3. Here we try to map these metrics based on their effectiveness collected from this SLR study.

Results for the evaluation metrics of RTS techniques are represented in Figure 8 with respect to cost, coverage, fault detection, and combination of these criteria. In the figure, x-axis represents evaluation metrics and y-axis represents effectiveness measure. In total, there are 11 metrics used to evaluate the results of RTS techniques obtained from the collected studies. Overall, 15 studies (31%) compare their results with previous run of the same application [S11, S35, S4, S18, S38, S43, S33, S17, S12, S37, S32, S22, S4, S39, S46], followed by *retest-all*, which is used in 9 studies (19%) [S25, S36, S13, S16, S10, S29, S5, S6, S44], precision, which is used in 8 studies (17%) [S1, S14, S18, S42, S13, S3, S7, S46], multi-objective metrics were reported in 5 studies (10%) [S23, S19, S24, S31, S34], and F-Measure, which is a combination of precision and recall, is used in 4 empirical studies (10%) [S27, S15, S26, S40].

Recall [S1, S40, S21, S46] is used in 4 primary studies (8%), inclusiveness [S2, S8, S42] is used in 3 primary studies (6%), APFD [S29, S41, S47] is used in 3 studies (6%), AFDR [S20, S28] is used in 2 studies (4%), and accuracy is used in 2 (4%) studies. The FDR [S30, S47] is used in 2 (4%) studies from primary studies under analysis. Few studies used more than one evaluation metric in order to compare different techniques of test case selection with reduction or prioritization techniques. We group evaluation metrics based on their effectiveness measures. Some studies used more than one metric to evaluate their results. There is a need to mention all metrics used in the studies. For example, study [S46] used precision, recall, and accuracy for its analysis; study [S47] used APFD and AFDR for its comparison with prioritization technique under analysis; study [S42] used inclusiveness and precision; and study [S1] used precision and recall to evaluate its results.

The results showed that choice of evaluation metrics depends upon the complexity of RTS technique used and nature of data produced either using experiment or case study. The *retest-all* and comparison with previous versions needs no extra computation and a simple comparison with previous project data can produce the needed results. The coverage information and cost is measured by tools provided by different vendors are common mechanism of adequacy and are used with precision and recall metrics. Fault-based adequacy criteria needs some analysis before and after experiments and also needs some historical information about test cases behaviors or fault history. Multi-objective metrics (S19, S23, S24, S31, S34) are used with machine-learning RTS techniques. They required the most complex calculations compared to other metrics.

6. DISCUSSION

Via the literature, it is found that two ideas contributed to the development of RTS techniques and their evaluation methods. The first idea is regression testing with test cases classification [Leung and White 1989] and the second idea is safe regression testing [Rothermel and Harrold 1994]. These studies provide the theoretical basis for evaluation of RTS techniques as well as their evaluation frameworks. But, with the increase in the complexity of software systems, testing environments, and frameworks, it is observed that with proper statistical backgrounds, these classic methods can still be effective. Many new trends (e.g., continuous development, continuous integration, agile and cloud-based systems) pose different challenges to regression testing. Mining and learning techniques are mainly used by researchers in this field of study, yet these methods have several drawbacks like small size datasets and high computational cost made it unattractive to industry players.

The second strong belief in regression testing is coverage based adequacy criteria. Coverage is used as the proxy for evaluating the quality of test suites. Coverage is a good indicator of test suite completeness in structural programs. But it is observed that coverage alone is not sufficient for test suite adequacy evaluation and insufficient measure for test suite quality measurement in emerging software domain. The high coverage also needs more number of test suites as well as consumes high costs. It seems that mutation based regression testing is gradually replacing the coverage measures. Mutation-based analysis on well-thought statistical grounds can reflect the quality of test suites better compared to static coverage measures. But there is still a gap between the theory and practice in mutation testing area. The ability of mutation analysis tools differ according to different quality measures used.

Based on this SLR, we recommend that researchers, practitioners, and engineers follow these guidelines if they intend to conduct regression test case selection:

1. Coverage, cost, or fault detection alone are not sufficient measures to assess the quality of test suites. They may be considered in some proportion with each other.
2. Coverage granularity levels should be an effective measure for test suites quality. But it is evident from the studies reviewed that higher coverage rates did not assure high fault detection ability of test suites.
3. RTS experimental studies must be designed to uncover fault with the cost of regression testing instead of fulfilling some coverage adequacy levels.
4. Testing artifact repositories, open source software with test suites, and supporting documents create positive impacts on regression test experiments.
5. There are a number of studies and body of knowledge for testing experimentation, and there must be a proper consideration for regression test experiment context, hypothesis, design, analysis method, threats to validity, data analysis, presentation, and results with conclusion.
6. It is also important to consider the accuracy, relevance, and impact of the domain under study.
7. The empirical studies provide a mechanism to relate controlled experiments with real world problems. So it is necessary to focus on measurable aspects of the regression testing, test case selection, prioritization, and reduction.
8. Based on this literature review, it is recommended that future studies need to be conducted using proper statistical methods.
9. An empirical study is credible when its results are reproducible. Thus, it is necessary to properly document the studies.
10. It seems that mutation testing methods are gradually replacing real fault and fault seeding techniques. This is because mutation testing methods can be used to effectively assess classic RTS techniques (e.g., Slicing, Firewall, Graph-Based Techniques).
11. There are also many indicators found in this SLR that mutation score may replace coverage adequacy criteria.
12. Software testing frameworks impose several limitations on experimental designs. It is suggested that future experiments be extended beyond these testing frameworks.

7. THREATS TO VALIDITY

For this SLR, the relevant threats to its validity may include incomplete selection of empirical studies, inaccurate data extraction, and unbiased selection strategy.

7.1. Selection of Publications

We have presented the research method in detail in Section 4. The SLR has a general guideline for the selection of relevant studies. In Section 4.5, we presented the strategy used to select the relevant studies. However, there is still a possibility that some relevant studies have been overlooked. The main reason for this is the existence of gray literature such as technical reports and PhD thesis. In this case, this literature is important, if the authors report the complete studies, but most of the time they are briefly reported. In this SLR, we did not include PhD theses and technical reports.

The second possible issue is the difficulty of finding appropriate search strings. In Section 4.3 and Section 4.4, justifications were given for the process of selection of repositories and search strings used to find relevant studies to be used in this SLR.

However, there may be some articles that may use some other keywords. We refine our search strings several times to identify the most relevant studies because some articles that are available in the references of some studies were missing from our search results. The studies were published in different sources, but we try our search query with most relevant data sources having a maximum probability to return to most relevant studies.

7.2. Inaccurate Data Extraction

Incomplete data extraction from the collected studies is the second possible reason for inaccurate results in this SLR. This may be due to unsystematic data extraction or invalid classification of data. First, we try to reduce the possibility of inaccurate data extraction by focusing on the data elements collected from the selected studies represented in Section 4.6. Second, all data extracted were reviewed three times. Empirical studies are focused and special type of experimental studies. Their experimental design and experimental process makes it possible to collect the data objectively.

7.3. Quality Assessment Problem

Quality assessment of selected studies is also another problem which may lead to inaccurate results. It is not easy to answer RQ 3 due to the nature of the measures needed to answer this question. For this purpose, we establish a framework presented in Section 3. This framework helps the researchers to assess the empirical studies for selection as well as to ensure their required quality assessment subjectively. We also emphasize that this is the minimum criteria to design and assess the empirical studies for regression test case selection.

8. CONCLUSION

We have presented results from our systematic review of empirical studies on RTS. This SLR is focused on the effectiveness of RTS techniques. Mainly, three (3) research questions were developed by introducing a framework. The findings, based on these research questions, are as follows:

(RQ1) In total, 47 studies were chosen from 724 studies obtained from the literature on RTS topic, which measure cost, cost-effectiveness, and/or effectiveness between 2007 and 2015. There were 5 distinct families of methods used namely Mining and Learning, Model-Based Testing (MBT), Program Slicing, Control Flow Graph (CFG), and Oracle-Based RTS identified within RTS techniques. These families were classified based on their operational procedure, input and output method, and solution similarities. There are several differences between the techniques used by each family in terms of context and environment, but, in general, they are used to solve the same kinds of problems. Mining and Learning method obtained better attention from researchers during this time span and is used in 38% of the selected studies. Genetic Algorithm is used in most experiments compared to other methods such as Fuzzy logic, Practical Optimization Swarm, and heuristic algorithms within the mining and learning technique family. The emerging crossroad from these RTS techniques is Test Case Profiling with historical data analysis to enhance effectiveness. The 24% studies conducted on a unit level testing to keep the problem simple and manageable within given context. There is need to expand the level of testing with the empirical evaluation of RTS optimization methods. Java programming language is reported in 28% studies. It shows that Java is the most accepted environment and object oriented solutions is more popular than structured solutions. Majority of the studies do not target any specific fault type but they used some structural coverage as their objectives. It is also observed that mutation score is starting to replace coverage criteria. There is a need to investigate RTS techniques with real faults in software applications using measurable characteristics of datasets.

(RQ2) We observed that 65% of the studies selected were case studies with small-size datasets. The use of industrial artifacts are still not common in testing case selection experiments. This factor also makes techniques applicability limited and no evidence available if they may or may not be used in large industrial applications. The studies conducted are not mature, and their findings cannot be generalized because only 37% of the studies used publicly available datasets, meaning that they can be reproduced by other researchers. The design guidelines for experimental setup and empirical evaluation guidelines are also compromised; only 32% of the studies explained their limitations and threats to validity. It is also observed that only 41% of the studies frame research questions to set up the empirical evaluation. It is important for studies, especially in the software engineering and software testing fields, to follow the proper experimental guidelines and using suitable statistical methods to ensure the validity of their studies.

(RQ3) The focus of this third and last research question was to assess the effectiveness of existing RTS techniques. The effectiveness was measured based on cost measure, coverage-based, and fault-based adequacy criteria. Of the 47 studies, 30 studies (63%) measured the cost of RTS technique using only one single parameter. It is found that 7 different cost measures were used in the studies. The most common cost measure used is execution time of the test suite that was used in 13 studies. The multi-objective adequacy criteria consist of a combination of these three adequacy criteria getting the focus of researchers and experimental studies in regression test case selection. The validation of these results is also equally important for empirical studies conducted in software testing in general and regression test case selection in particular. We found that 15 studies compared their results with previous versions of the same test suite execution. It is also observed that the use of a single metric (either cost, coverage, or fault criteria) is not sufficient for measuring all aspects of test case selection techniques. Furthermore, there is no direct relationship found between effectiveness based on the cost, coverage, or fault detection ability. The definition of effectiveness varies among all studies selected. The cost is measured from different perspectives and conflicting definitions. A measure like reduction in size is considered as effective, but it does not guarantee a reduction in the execution cost of the test suite. Similar trends is also found in coverage and fault detection ability.

9. FUTURE WORK

Potential future work for the researchers in this field may include:

(1) Perform research focusing on trade-offs between current RTS techniques (e.g., traditional techniques) with mining and learning or test oracles (i.e., techniques with statistical origins).
(2) Design a general purpose evaluation technique to be used for the whole RTS testing process, because we were unable to find a single fit-for-all technique to be used for this purpose.
(3) More works are required to design and conduct empirical studies, because the current design guidelines are not mature and can be further improved.
(4) Single adequacy criterion is not sufficient, thus investigations should be made to multi-criteria cases and performed with proper statistical methods used.
(5) There is need to determine the relationships between cost, coverage, and fault-detection abilities of test suites.
(6) Evaluation metrics should be implemented with sufficient mathematical and statistical details, instead of only making comparison with previous studies' results, because in many cases the testing context may differ although implemented on the same application.

ACKNOWLEDGMENTS

We thank Universiti Teknologi Malaysia and Ministry of Science, Technology and Innovation (MOSTI) Malaysia (Vot No: 4S113).

REFERENCES

J. Anderson, S. Salem, and H. Do. 2014. Improving the effectiveness of test suite through mining historical data. In *Proceedings of the 11th Working Conference on Mining Software Repositories.* ACM, 142–151.

J. H. Andrews, L. C. Briand, Y. Labiche, and A. S. Namin. 2006. Using mutation analysis for assessing and comparing testing coverage criteria. *IEEE Trans. Software Eng.* 32, 608–624.

M. A. Askarunisa, M. L. Shanmugapriya, and D. N. Ramaraj. 2010. Cost and coverage metrics for measuring the effectiveness of test case prioritization techniques. *INFOCOMP J. Comput. Sci.* 9, 43–52.

A. Assis Lobo De Oliveira, C. Gonyalves Camilo-Junior, and A. M. Vincenzi. 2013. A coevolutionary algorithm to automatic test case selection and mutant in mutation testing. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'13).* IEEE, 829–836.

B. Beizer. 1995. *Black-Box Testing: Techniques for Functional Testing of Software and Systems*, John Wiley & Sons, Inc.

D. Binkley. 1995. Reducing the cost of regression testing by semantics guided test case selection. In *Proceedings of the International Conference on Software Maintenance.* IEEE, 251–260.

S. Biswas, R. Mall, M. Satpathy, and S. Sukumaran. 2011a. Regression test selection techniques: A survey. *Informat.: Int. J. Comput. Informat.* 35, 289–321.

S. Biswas, R. Mall, M. Satpathy, and S. Sukumaran. 2011b. Regression test selection techniques: A survey. *Informatica* 35.

P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Software* 80, 571–583.

X. Cai and M. R. Lyu. 2005. The effect of code coverage on fault detection under different testing profiles. *ACM SIGSOFT Software Eng. Notes* 30, 1–7.

E. G. Cartaxo, P. D. Machado, and F. G. O. Neto. 2011. On the use of a similarity function for test case selection in the context of model-based testing. *Software Test. Verificat. Reliabil.* 21, 75–100.

S. Chen, Z. Chen, Z. Zhao, B. Xu, and Y. Feng. 2011a. Using semi-supervised clustering to improve regression test selection techniques. In *Proceedings of the IEEE 4th International Conference on Software Testing, Verification and Validation (ICST'11).* IEEE, 1–10.

Z. Chen, Y. Duan, Z. Zhao, B. Xu, and J. Qian. 2011b. Using program slicing to improve the efficiency and effectiveness of cluster test selection. *Int. J. Software Eng. Knowl. Eng.* 21, 759–777.

P. K. Chittimalli and M. J. Harrold. 2009. Recomputing coverage information to assist regression testing. *IEEE Trans. Software Eng.* 35, 452–469.

H. Cibulski and A. Yehudai. 2011. Regression test selection techniques for test-driven development. In *Proceedings of the 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW'11).* IEEE, 115–124.

C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont. 2002. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Springer.

L. S. De Souza, P. B. De Miranda, R. B. Prudencio, and F. De Barros. 2011. A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort. In *Proceedings of the23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI'11).* IEEE, 245–252.

L. S. De Souza, R. B. Prudencio, and D. A. Barros. 2014a. A hybrid binary multi-objective particle swarm optimization with local search for test case selection. In *Proceedings of the 2014 Brazilian Conference on Intelligent Systems (BRACIS'14)*, 2014a. IEEE, 414–419.

L. S. De Souza, R. B. Prudencio, and F. D. Barros. 2014b. A comparison study of binary multi-objective particle swarm optimization approaches for test case selection. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC'14).* IEEE, 2164–2171.

L. S. De Souza, R. B. Prudêncio, F. D. A. Barros, and E. H. D. S. Aranha. 2013. Search based constrained test case selection using execution effort. *Expert Syst. Appl.* 40, 4887–4896.

K. Deb. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons.

M. E. Delamaro and J. Offutt. 2014. Assessing the influence of multiple test case selection on mutation experiments. In *Proceedings of the 2014 IEEE 7th International Conference on Software Testing, Verification and Validation Workshops (ICSTW'14).* IEEE, 171–175.

H. Do and G. Rothermel. 2006. An empirical study of regression testing techniques incorporating context and lifetime factors and improved cost-benefit models. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering.* ACM, 141–151.

T. Dyba, B. A. Kitchenham, and M. Jorgensen. 2005. Evidence-based software engineering for practitioners. *IEEE Software*, 22, 58–65.

I. Eee. 1990. Standard G lossary of softwareengineering terminology. *IEEE Software Eng. Stand. Cll ect. IEEE*, 610.12–190.

W. S. A. El-Hamid, S. S. El-Etriby, and M. M. Hadhoud. 2010. Regression test selection technique for multi-programming language. In *Proceedings of the 7th International Conference on Informatics and Systems (INFOS'10).* IEEE, 1–5.

S. Elbaum, P. Kallakuri, A. Malishevsky, G. Rothermel, and S. Kanduri. 2003. Understanding the effects of changes on the cost-effectiveness of regression testing techniques. *Software Test. Verificat. Reliabil.* 13, 65–83.

S. Elbaum, A. Malishevsky, and G. Rothermel. 2001. Incorporating varying test costs and fault severities into test case prioritization. In *Proceedings of the 23rd International Conference on Software Engineering.* IEEE Computer Society, 329–338.

S. Elbaum, A. G. Malishevsky, and G. Rothermel. 2002. Test case prioritization: A family of empirical studies. *IEEE Trans. Software Eng.* 28, 159–182.

E. Engström, P. Runeson, and M. Skoglund. 2010. A systematic review on regression test selection techniques. *Informat. Software Technol.* 52, 14–30.

E. Engström, M. Skoglund, and P. Runeson. 2008. Empirical evaluations of regression test selection techniques: A systematic review. In *Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement.* ACM, 22–31.

K. F. Fischer. 1977. A test case selection method for the validation of software maintenance modifications. *Proceedings of COMPSAC*, 1977. 421–426.

E. Fourneret, J. Cantenot, F. Bouquet, B. Legeard, and J. Botella. 2014. Setgam: Generalized technique for regression testing based on UML/OCL models. In *Proceedings of the 8th International Conference on Software Security and Reliability (SERE'14)*, 2014. IEEE, 147–156.

M. Gligoric, L. Eloussi, and D. Marinov. 2015. Practical regression test selection with dynamic file dependencies. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis.* ACM, 211–222.

M. Gligoric, A. Groce, C. Zhang, R. Sharma, M. A. Alipour, and D. Marinov. 2014. Guidelines for coverage-based comparisons of non-adequate test suites. *Space* 6, 1, 142.

J. E. González, N. Juristo, and S. Vegas. 2014. A systematic mapping study on testing technique experiments: has the situation changed since 2000? In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2014. ACM, 3.

T. L. Graves, M. J. Harrold, J. M. Kim, A. Porter, and G. Rothermel. 2001. An empirical study of regression test selection techniques. *ACM Trans. Software Eng. Methodol (TOSEM)* 10, 184–208.

F. Haftmann, D. Kossmann, and E. Lo. 2007. A framework for efficient regression tests on database applications. *VLDB J.: Int. J. Very Large Data Bases* 16, 145–164.

M. Harman and N. Alshahwan. 2008. Automated session data repair for web application regression testing. In *Proceedings of the 1st International Conference on Software Testing, Verification, and Validation.* IEEE, 298–307.

H. Hemmati, A. Arcuri, and L. Briand. 2010a. Reducing the cost of model-based testing through test case diversity. In *Testing Software and Systems.* Springer.

H. Hemmati, A. Arcuri, and L. Briand. 2011. Empirical investigation of the effects of test suite properties on similarity-based test case selection. In *Proceedings of the IEEE 4th International Conference on Software Testing, Verification and Validation (ICST'11).* IEEE, 327–336.

H. Hemmati and L. Briand. 2010. An industrial investigation of similarity measures for model-based test case selection. In *Proceedings of the IEEE 21st International Symposium on Software Reliability Engineering (ISSRE'10)* IEEE, 141–150.

H. Hemmati, L. Briand, A. Arcuri, and S. Ali. 2010b. An enhanced test case selection approach for model-based testing: An industrial case study. In *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering.* ACM, 267–276.

K. Hla, Y. Choi, and J. S. Park. 2008. Applying particle swarm optimization to prioritizing test cases for embedded real time software retesting. In *Proceedings of the IEEE 8th International Conference on Computer and Information Technology Workshops.* IEEE, 527–532.

S. Huang, Y. Chen, J. Zhu, Z. J. Li, and H. F. Tan. 2009. An optimized change-driven regression testing selection strategy for binary Java applications. In *Proceedings of the 2009 ACM Symposium on Applied Computing.* ACM, 558–565.

S. Huang, Z. J. Li, J. Zhu, Y. Xiao, and W. Wang. 2011. A novel approach to regression test selection for J2EE applications. In *Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM'11).* IEEE, 13–22.

IEEE-STD-610. 12-1990. 1990. IEEE standard glossary of software engineering terminology (IEEE Std 610.12-1990). IEEE Computer Society, Los Alamitos. CA.

S. A. Infrastructure. 2016. *SIR.* Retrieved from http://sir.unl.edu/portal/index.php.

L. Inozemtseva and R. Holmes. 2014. Coverage is not strongly correlated with test suite effectiveness. In *Proceedings of the 36th International Conference on Software Engineering.* ACM, 435–445.

M. Z. Z. Iqbal, Z. Malik, and M. Riebisch. 2010. A model-based regression testing approach for evolving software systems with flexible tool support. In *Proceedings of the 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS'10).* IEEE, 41–49.

L. Jiang and Z. Su. 2007. Context-aware statistical debugging: from bug predictors to faulty control flow paths. In *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*. ACM, 184–193.

D. S. Johnson. 2002. A theoretician's guide to the experimental analysis of algorithms. *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, 59, 215–250.

B. Kitchenham. 2004. Procedures for performing systematic reviews. Keele University, Keele, UK. 33, 1–26.

B. A. Kitchenham, T. Dyba, and M. Jorgensen. 2004. Evidence-based software engineering. In *Proceedings of the 26th International Conference on Software Engineering.* IEEE Computer Society, 273–281.

B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg. 2002. Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Software Eng.* 28, 721–734.

M. Kumar, A. Sharma, and R. Kumar. 2013. Fuzzy entropy-based framework for multi-faceted test case classification and selection: An empirical study. *IET Software* 8, 103–112.

M. Kumar, A. Sharma, and R. Kumar. 2015. An empirical evaluation of a three-tier conduit framework for multifaceted test case classification and selection using fuzzy-ant colony optimisation approach. *Software: Pract. Exp.* 45, 949–971.

D. C. Kung, C. H. Liu, and P. Hsia. 2000. An object-oriented web test model for testing web applications. In *Proceedings of the 1st Asia-Pacific Conference on Quality Software.* IEEE, 111–120.

H. K. Leung and L. White. 1989. Insights into regression testing [software testing]. In *Proceedings of the 1989 Conference on Software Maintenance.* IEEE, 60–69.

H. K. Leung and L. White. 1991. A cost model to compare regression test strategies. In *Proceedings of the 1991 Conference on Software Maintenance.* IEEE, 201–208.

W. E. Lewis. 2008. *Software Testing and Continuous Quality Improvement*, CRC Press, Boca Raton, FL.

W. E. Lewis. 2016. *Software Testing and Continuous Quality Improvement*, CRC Press, Boca Raton, FL.

B. Li, D. Qiu, H. Leung, and D. Wang. 2012. Automatic test case selection for regression testing of composite service based on extensible BPEL flow graph. *J. Syst. Software* 85, 1300–1324.

Y. D. Lin, C. H. Chou, Y. C. Lai, T. Y. Huang, S. Chung, J. T. Hung, and F. C. Lin. 2012. Test coverage optimization for large code problems. *J. Syst. Software* 85, 16–27.

N. Mansour, H. Takkoush, and A. Nehme. 2011. UML-based regression testing for OO software. *J. Software Maint. Evol.: Res. Pract.* 23, 51–68.

A. Memon, A. Nagarajan, and Q. Xie. 2005. Automating regression testing for evolving GUI software. *J. Software Maint. Evol.: Res. Pract.* 17, 27–64.

A. M. Memon. 2008. Automatically repairing event sequence-based GUI test suites for regression testing. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 18, 4.

A. M. Memon and M. L. Soffa. 2003. Regression testing of GUIs. *ACM SIGSOFT Software Engineering Notes* 28, 118–127.

S. Mirarab, S. Akhlaghi, and L. Tahvildari. 2012a. Size-constrained regression test case selection using multicriteria optimization. *IEEE Trans. Software Eng.* 38, 936–956.

S. Mirarab, S. Akhlaghi, and L. Tahvildari. 2012b. Size-constrained regression test case selection using multicriteria optimization. *IEEE Trans. Software Eng.* 38, 936–956.

J. D. Musa. 1993. Operational profiles in software-reliability engineering. *IEEE Software* 10, 14–32.

R. Nagar, A. Kumar, S. Kumar, and A. S. Baghel. 2014. Implementing test case selection and reduction techniques using meta-heuristics. In *Proceedings of the 5th International Conference on Confluence The Next Generation Information Technology Summit (Confluence'14)*. IEEE, 837–842.

A. S. Namin and J. H. Andrews. 2009. The influence of size and coverage on test suite effectiveness. *Proceedings of the 18th International Symposium on Software Testing and Analysis.* ACM, 57–68.

A. Nanda, S. Mani, S. Sinha, M. J. Harrold, and A. Orso. 2011. Regression testing in the presence of non-code changes. In *Proceedings of the IEEE 4th International Conference on Software Testing, Verification and Validation (ICST'11)*. IEEE, 21–30.

D. D. Nardo, N. Alshahwan, L. Briand, and Y. Labiche. 2015. Coverage-based regression test case selection, minimization and prioritization: A case study on an industrial system. *Software Test. Verificat. Reliabil.* 25, 371–396.

A. Orso and G. Rothermel. 2014. Software testing: A research travelogue (2000–2014). In *Proceedings on the Future of Software Engineering.* ACM, 117–132.

A. Orso, N. Shi, and M. J. Harrold. 2004. Scaling regression testing to large software systems. *ACM SIGSOFT Software Engineering Notes*, 2004. ACM, 241–251.

T. J. Ostrand, E. J. Weyuker, and R. M. Bell. 2005. Predicting the location and number of faults in large software systems. *IEEE Trans. Software Eng.* 31, 340–355.

Y. Pang, X. Xue, and A. S. Namin. 2013. Identifying effective test cases through k-means clustering for enhancing regression testing. In *Proceedings of the 12th International Conference on Machine Learning and Applications (ICMLA'13)*. IEEE, 78–83.

A. Panichella, R. Oliveto, M. D. Penta, and A. De Lucia. 2015. Improving multi-objective test case selection by injecting diversity in genetic algorithms. *IEEE Trans. Software Eng.* 41, 358–383.

A. Pasala, Y. Lew Yaw Fung, F. Akladios, G. Appala Raju, and R. P. Gorthi. 2008. Selection of regression test suite to validate software applications upon deployment of upgrades. In *Proceedings of the 19th Australian Conference on Software Engineering (ASWEC'08)*. IEEE, 130–138.

S. Poulding, P. Emberson, I. Bate, and J. Clark. 2007. An efficient experimental methodology for configuring search-based design algorithms. In *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*. IEEE, 53–62.

X. Qu, M. B. Cohen, and G. Rothermel. 2008. Configuration-aware regression testing: an empirical study of sampling and prioritization. In *Proceedings of the 2008 International Symposium on Software Testing and Analysis.* ACM, 75–86.

N. Rachatasumrit and M. Kim. 2012. An empirical investigation into the impact of refactoring on regression testing. In *Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM'12)*. IEEE, 357–366.

D. Roest, A. Mesbah, and A. Van Deursen. 2010. Regression testing ajax applications: Coping with dynamism. In *Proceedings of the 3rd International Conference on Software Testing, Verification and Validation (ICST'10)*. IEEE, 127–136.

E. Rogstad, L. Briand, and R. Torkar. 2013. Test case selection for black-box regression testing of database applications. *Informat. Software Technol.* 55, 1781–1795.

D. S. Rosenblum and E. J. Weyuker. 1997. Using coverage information to predict the cost-effectiveness of regression testing strategies. *IEEE Trans. Software Engineering* 23, 146–156.

G. Rothermel. 1996. *Efficient, Effective Regression Testing Using Safe Test Selection Techniques.* Clemson University.

G. Rothermel and M. J. Harrold. 1994. A framework for evaluating regression test selection techniques. In *Proceedings of the 16th International Conference on Software Engineering (ICSE-16)*. IEEE, 201–210.

G. Rothermel and M. J. Harrold. 1996. Analyzing regression test selection techniques. *IEEE Trans. Software Eng.* 22, 529–551.

G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong. 1998. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In *Proceedings of the 1998 Proceedings of the International Conference on Software Maintenance.* IEEE, 34–43.

G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. 2001. Prioritizing test cases for regression testing. *IEEE Trans. Software Engineering*, 27, 929–948.

P. Sapna and H. Mohanty. 2010. Clustering test cases to achieve effective test selection. *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*, 2010. ACM, 15.

W. Schütz. 1994. Fundamental issues in testing distributed real-time systems. *Real-Time Syst.* 7, 129–157.

W. R. Shadish, T. D. Cook, and D. T. Campbell. 2002. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Wadsworth Cengage Learning.

A. Shi, T. Yung, A. Gyori, and D. Marinov. 2015. Comparing and combining test-suite reduction and regression test selection. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering.* ACM, 237–247.

Y. Singh, A. Kaur, and B. Suri. 2010. A hybrid approach for regression testing in interprocedural program. *JIPS*, 6, 21–32.

C. Tao, B. Li, X. Sun, and C. Zhang. 2010a. An approach to regression test selection based on hierarchical slicing technique. In *Proceedings of the IEEE 34th Annual Computer Software and Applications Conference Workshops (COMPSACW'10).* IEEE, 347–352.

C. Tao, B. Li, X. Sun, and Y. Zhou. 2010b. A hierarchical model for regression test selection and cost analysis of Java programs. In *Proceedings of the 17th Asia Pacific Software Engineering Conference (APSEC'10).* IEEE, 290–299.

W. T. Tsai, X. Zhou, R. A. Paul, Y. Chen, and X. Bai. 2009. A coverage relationship model for test case selection and ranking for multi-version software. In *High Assurance Services Computing.* Springer.

R. Victor. 2003. Iterative and incremental development: A brief history. *IEEE Computer Society*, 47–56.

L. White. 1989. Insights IntoRegressionTesting. In *Proceedings of the Conference on Software Maintenance.* IEEE Computer Society Press, 60–69.

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

C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. 2012. *Experimentation in Software Engineering*, Springer Science & Business Media.

W. E. Wong, J. R. Horgan, A. P. Mathur, and A. Pasquini. 1997. Test set size minimization and fault detection effectiveness: A case study in a space application. In *Proceedings on the 21st Annual International Computer Software and Applications Conference (COMPSAC'97).* IEEE, 522–528.

G. Xu and A. Rountev. 2007. Regression test selection for AspectJ software. In *Proceedings of the 29th International Conference on Software Engineering (ICSE'07).* IEEE, 65–74.

L. Xu, B. Xu, Z. Chen, J. Jiang, and H. Chen. 2003. Regression testing for web applications based on slicing. In *Proceedings on the 27th Annual International Computer Software and Applications Conference (COMPSAC'03).* IEEE, 652–656.

Z. Xu, K. Gao, T. M. Khoshgoftaar, and N. Seliya. 2014. System regression test planning with a fuzzy expert system. *Informat. Sci.* 259, 532–543.

Z. Xu, Y. Liu, and K. Gao. 2013. A novel fuzzy classification to enhance software regression testing. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM'13).* IEEE, 53–58.

R. K. Yin. 2003. Case study research design and methods third edition. *Applied Social Research Methods Series*, 5.

S. Yoo and M. Harman. 2007. Pareto efficient multi-objective test case selection. In *Proceedings of the 2007 International Symposium on Software Testing and Analysis.* ACM, 140–150.

S. Yoo and M. Harman. 2012. Regression testing minimization, selection and prioritization: A survey. *Software Test. Verificat. Reliabil.* 22, 67–120.

L. Yu, L. Xu, and W. T. Tsai. 2010. Time-constrained test selection for regression testing. *Advanced Data Mining and Applications.* Springer.

T. Yu, X. Qu, M. Acharya, and G. Rothermel. 2013. Oracle-based regression test selection. In *Proceedings of the IEEE 6th International Conference on Software Testing, Verification and Validation (ICST'13).* IEEE, 292–301.

L. Zhang, S. S. Hou, C. Guo, T. Xie, and H. Mei. 2009. Time-aware test-case prioritization using integer linear programming. In *Proceedings of the 18th International Symposium on Software Testing and Analysis.* ACM, 213–224.

J. Zheng, L. Williams, B. Robinson, and K. Smiley. 2007. Regression test selection for black-box dynamic link library components. In *Proceedings of the 2nd International Workshop on Incorporating COTS Software Into Software Systems: Tools and Techniques.* IEEE Computer Society, 9.

Received February 2016; revised November 2016; accepted February 2017