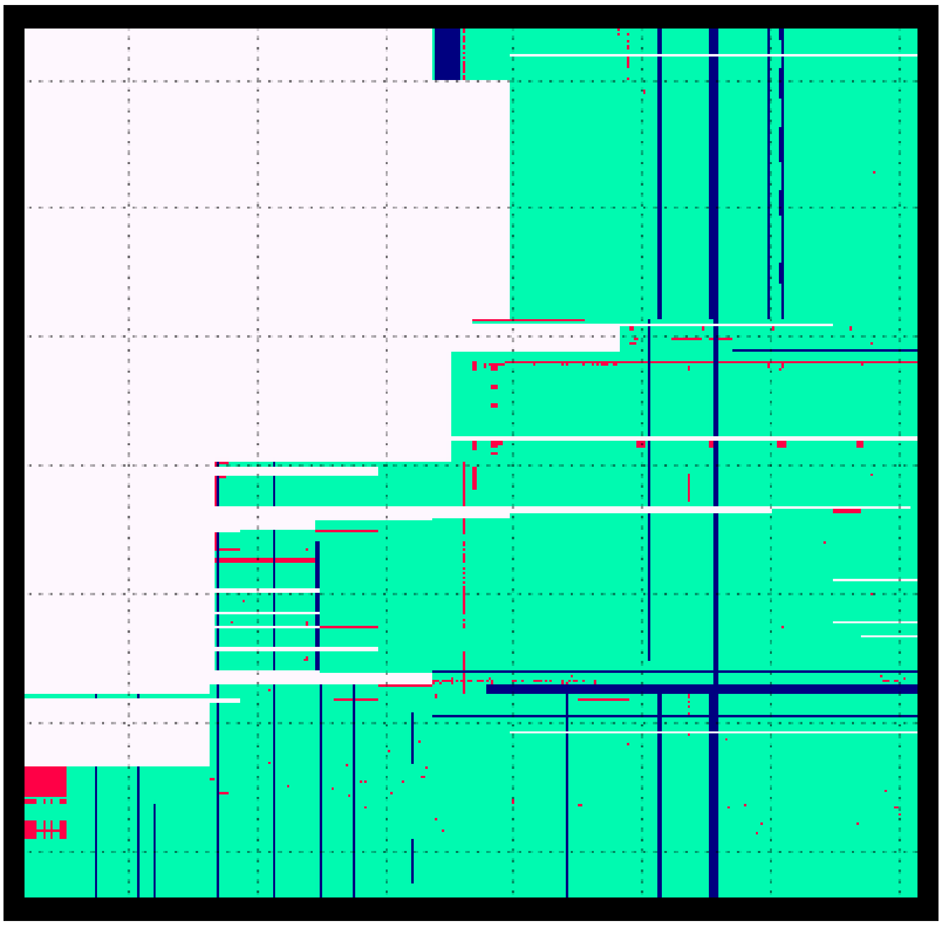
CHALMERS



A Framework for Evaluating Regression Test Selection Techniques in Industry

Master of Science Thesis Software Engineering and Technology

ALEX AUGUSTSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, April 2012

A Framework for Evaluating Regression Test Selection Techniques in Industry

ALEX AUGUSTSSON

© ALEX AUGUSTSSON, April 2012.

Examiner: Associate Professor Miroslav Staron
Supervisor: Associate Professor Robert Feldt

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

The cover page picture is a visualization of an excerpt of regression testing status data where the x-axis represents execution sessions, the y-axis represents test cases and the colors represent the test case status (fail/pass/etcetera).

Department of Computer Science and Engineering
Göteborg, Sweden, April 2012

Acknowledgements

The author is thankful to Associate Professor Robert Feldt at the Dept. of Software Engineering, Chalmers Technical University and Dr. Perolof Bengtsson and Bertil Aspernäs at Ericsson in Karlskrona for their support.

A Framework for Evaluating Regression Test Selection Techniques in Industry

Alex Augustsson
MS Student, Software Engineering and Technology
Chalmers University of Technology
Göteborg, Sweden
alex.augustsson@gmail.com

***Abstract— Background*: Previous research in the area of regression testing has mainly focused on different techniques used to decrease the size of test suites. However, studies that compare the techniques in authentic industrial contexts are few. *Aim*: The aim of this paper is to introduce an efficient, purposeful framework meant to evaluate regression test selection techniques using only a limited selection of available information. *Method*: In order to evaluate and compare different regression testing techniques three realistic and important scenarios were recognized and a framework was developed. This was then utilized as a starting point for an evaluation case study which compared regression test selection techniques. Regression test data was collected from a software developing site within Ericsson. *Results*: The framework evaluation showed that a well-supported decision could be made regarding which regression testing technique a software development organization should use. The comparative case study also showed that, compared to a random selection, a technique based on historical test data improved the fail detection. *Conclusions*: The contribution of this paper is the framework which can be used as a basis for further research as well as aid practitioners in the analysis and evaluation of regression test selection techniques.**

***Keywords*— regression testing; evaluation framework; industrial context; regression test selection; historical test data; information-constrained**;

I. Introduction

Regression testing is conducted to find faults, or to assure that faults does not exist, in currently existing software when adding new code or modifying existing code. A typical approach to regression testing is to retest all test cases (retest-all) [1] and to allow the testing to be carried out continuously during iterative development [2]. This is an expensive activity as the amount of code increases continuously, resulting in a growing number of test cases to be executed. Studies indicate that regression testing can stand for as much as 80% of the total testing cost [3]. This amount can add up to as much as 50% of the total cost of the software product [4]. Also, the execution time of the test cases can impose a bottle-neck when developing large software systems which could ultimately result in quality issues.

Decreasing the number of executed test cases is thus one possible way of reducing costs and execution time. In order to accomplish this several different techniques has been proposed. Yoo and Harman divide these techniques into three categories [5]. To permanently reduce the test suite the Test Suite Minimization (TSM) approach is used. The second category, selecting a subset of test cases to be executed for a given version of the software, is called Regression Test Selection (RTS). The third category, the Test Case Prioritization (TCP) technique, is not primarily about decreasing the number of executed test cases. Instead it prioritizes the test cases by fault detection likelihood. However, a TCP can be considered as an RTS. For instance, a test case selection can be made by setting a threshold value (e.g. 20%) choosing only the most prioritized test cases for execution.

Most regression techniques are based on the analysis of code [5, 6]. However, in industry, the use of such techniques is not always feasible due to the lack of access to the full code [2]. Also, the techniques require certain data which can be too expensive to both collect and maintain. One example is the traceability between code and test cases. It might also be the case that techniques which require more information demand more advanced knowledge, thus making practitioners disregard the techniques due to the learning threshold.

Research indicates that there is no basis for choosing a superior RTS technique [6, 23]. Moreover, the industry does not seem to have either a method or a practice supporting their choice of technique [3, 6]. In practice it is thinkable that the knowledge and experience of the developers determine the selection and that it is not a systematic approach [3].

In existing research the main focus has been on the selection methods rather than on how to evaluate and compare them. This presents a problem. Before selections and evaluations can be discussed and optimized, independently defined measures on how to evaluate different selection schemes must be defined.

This paper aims to examine the possibility of creating a framework which uses a minimal selection of information, extracted from historical test data, to evaluate different RTS techniques. This data is often a byproduct of the testing and it is often easily accessible. A prototype tool was developed in order to facilitate the collection and the preprocessing of data. It implemented different RTS techniques and evaluation schemes but it also supported regression test data analysis and the evaluation of the framework itself. The framework was developed through comparative studies of different RTS techniques connected to different real-life situations. These real-life situations are referred to as scenarios.

When creating the framework, data from a site within Ericsson AB was used. Said site develops services to mobile phone operating companies. During a time period of more than two years, data was collected from one distinct system consisting of millions of LOC.

The contribution of this paper is the proposed framework which consists of foundations and building blocks for analyses, visualizations, comparisons and evaluations of different RTS techniques using only a minimum selection of information. The introduced framework is intended to be simple and undemanding for practitioners to understand and implement. Furthermore the framework is required to support a multitude of different analyses since the amount of available data can vary during the lifecycle of a project.

This paper also aims to summarize the current state of art when it comes to evaluation of the regression testing applicable in industry. To accomplish this, the following research questions are meant to be answered:

RQ1. What empirical studies, with a large scale industrial perspective, have been conducted on evaluation of RTS techniques?

RQ2. How can RTS techniques be evaluated objectively?

RQ3. How can a framework, intended to evaluate different RTS techniques on realistic data, be created?

RQ4. How can RTS techniques be categorized based on their effectiveness?

This paper is structured as follows; Section II introduces the related work in this area. Section III presents the method and Section IV the different building blocks of the proposed framework. Section V presents a comparative study of different RTS techniques. Section VI contains the discussion of the results and in section VII the conclusion and further work are presented. This is followed by references.

II. RELATED WORK

The area of regression testing is well researched. Two extensive studies have been conducted, one regarding the area in full [5] and one focusing on papers regarding RTS [6].

A. *Empirical evaluations*

Comparative evaluation studies of regression testing techniques are limited, in particular those conducted in an industrial context. Most studies investigate small to medium size systems, leaving the question whether the result is applicable to larger systems unanswered.

There are, however, a couple of studies on evaluation of RTS techniques regarding large systems. Orso et al. [7] evaluates the use of a two-phase RTS technique on Java programs and compares it to a high-level firewall technique, an edge-level identification technique as well as retest-all.

The high-level firewall technique is, in turn, compared to a change-based selection technique presented by Skoglund and Runeson [8]. Skoglund and Runeson [8] also provide a large-scale industrial validation.

Another large-scale industrial study is presented by White et al. [9]. Two firewall approaches (procedural-design, additional data-paths) are compared to the high-level firewall technique previously mentioned. Furthermore, White and Robinson [10] compare the high-level firewall and the procedural-design firewall with an intuitive-based approach respectively.

B. *Evaluation frameworks*

Rothermel and Harrold present a framework for evaluation of regression test selections, introducing four categories of evaluation; inclusiveness, precision, efficiency and generality [11]. When comparing the most recent output from a test case to that from a previous execution it may have changed. Measuring how well a technique detects those test cases is called inclusiveness. The second category, precision, measures the capability to exclude test cases that do not produce a different output between executions. The third category, efficiency, provides the costs of the computations for the selection. And the last category, generality, provides an indication on how general the technique is; whether it can handle different programming languages, different degrees of complex code modifications or realistic testing applications.

C. *RTS technique categorization*

Categorization of RTS techniques has also been dealt with in research. Engström et al. [6] establish that RTS techniques could be categorized in various ways, such as: language applicability; which input is needed for different methods; the used approach; level of granularity regarding changes; different properties for methods such as safe/unsafe, minimizing/not minimizing, dataflow-coverage-based, ad hoc/random etcetera.

D. *Metrics*

Engström et al. summarize metrics used as evaluation criteria for RTS techniques [6]. Reduction of cost is commonly used as an evaluation criterion. To measure this, test suite reduction is mostly used. The total time, i.e. the time of test selection and test execution combined, is also used. Metrics for effectiveness are divided into measures relating to the number of failing test cases or the number of faults found from selected test cases.

One metric that is used as a de facto standard for TCP techniques is the Average Percentage Fault Detection (AFPD) [12]. It measures the rate with which the TCP techniques prioritize fault detecting test cases. Variations of this metric are presented, including cost [13] and failing test cases instead of faults [14].

Even though there are metrics regarded as standard-metrics, there is no conclusion made as to for which situation a specific metric is most suitable. It is stated that the use of a certain metric is based on the design of the study [15].

III. Method

The following section describes the industrial context and the methodology. The data is presented in detail and validity threats to the conducted research are identified.

A. *Industrial context*

The work involved in the research process was executed in collaboration with the development unit Revenue Management of Ericsson AB, located in Karlskrona, Sweden. Ericsson is a world-leading provider of telecommunication equipment and data communication systems. The Karlskrona division develops software systems for mobile communication. The system that was studied is an essential part of the mobile communication solutions.

Currently two versions of the product/system are maintained. Both versions include specific product customizations projects as well as integration projects alongside the main systems. Regression testing is carried out all nights throughout the week, with few exceptions, for every project and each part of the system.

The test cases are logically grouped according to existent communication interfaces. These groups are referred to as test objects. Data from each test case is collected and stored in a database.

B. *Study design*

The method used in this study is based on the Design Research paradigm [16, 17]. Fig. 1 describes the general process.

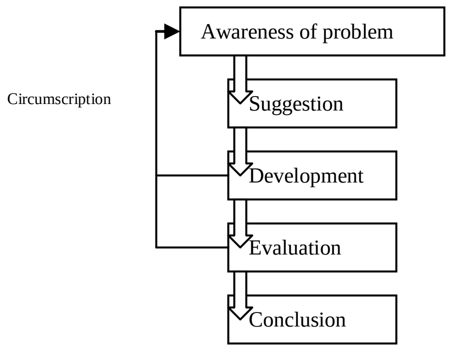


Figure 1. The general methodology of design research adapted by [17]

A tentative design was implemented in the development phase and in this particular case the following steps were taken:

1. Creating and defining fundamental concepts regarding the proposed framework.
2. Identifying regression testing scenarios relevant for industrial use.
3. Creating the framework.
4. Developing a tool suitable for analysis and evaluation of regression testing techniques.
5. Collecting and preprocessing data.

Following the problem statement and suggested solution the first step was to create taxonomy of different basic metrics for analysis and evaluation as well as the actual evaluations. Simultaneously, regression testing scenarios of industrial importance were identified. The combined work of the taxonomy creation and the scenario identification resulted in the framework's building blocks. The developed tool was, in turn, based on the framework. Data was then collected and used by the tool in order to analyze and evaluate different RTS techniques. Fig. 2 shows the steps in the development phase. The work within the development phase was conducted in an iterative way.

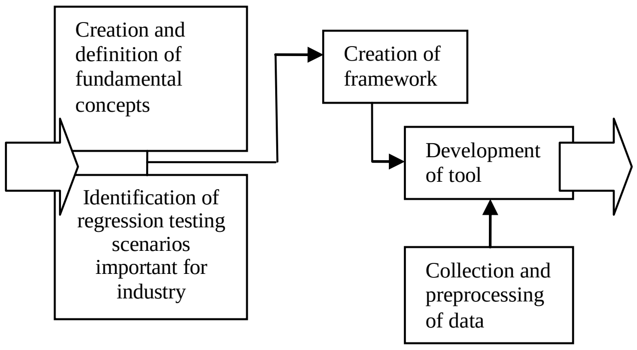

Creation and definition of fundamental concepts
Identification of regression testing scenarios important for industry
Creation of framework
Development of tool
Collection and preprocessing of data

Figure 2. Steps in the development phase.

The evaluation phase consisted of two stages. In the first stage the framework, and its usage when comparing RTS techniques, was described. The comparison mentioned above was realized as a case study where two regression testing techniques were set against each other. An industrial scenario was then chosen and a step-by-step walk-through of the parts in the framework was conducted in order to decide the best technique for this scenario. The second stage consisted of a broader review which introduced two more scenarios.

The conclusion phase included elaboration about the proposed framework and if it was fit for its purpose.

C. *Data*

The data used in this research was collected from one project and one version of the software, in this paper it is referred to as release A. It consisted of approximately 400 sessions which were executed between October 2010 and August 2011. Primarily the sessions were executed during the night i.e. once a day. In the beginning, roughly 2400 test cases were executed and in the end of the period this number had increased to almost 4100. This was also the maximum number of executed test cases during any session. In total, approximately 4600 distinct test cases were executed during all sessions.

The execution data was stored in a database along with test case descriptions. Data for the chosen project was extracted and preprocessed. Then, the data was presented in a matrix where each row represents a test case and each column represents a session. Each cell contains a symbol P (passed), F (failed), N (not applicable), 0 (null) or X (not alive). A test case is set to X in the sessions before it has come into existence and after it has seized to exist, whereas

N represents a manual removal of a test case. The intention of setting a test case to ‘not applicable’ is to permanently remove the test case the next session. When a test case is 0 it is either manually removed or the execution has halted. In the first situation there are primarily two possible scenarios; a test case can be in “quarantine” for a couple of sessions, for instance because it is not properly designed or, a test case can be removed permanently even though it has not been set to N previously.

Minimal selection of information, as mentioned earlier, suggests using only F, P and 0. The information used in this paper also includes N and X as stated. The use of N is an adaptation for the Ericsson case and the use of X is meant to separate the not-alive test cases from those referred to as null. This inclusion of N and X together with P, F and 0 is called realistic minimal selection of information. The suggested framework supports both approaches of information selection.

The collected data was used together with the implementation of the framework in order to analyze the data and RTS techniques as well as to develop and evaluate the framework.

D. *Validity threats*

In this study the validity threats are divided into four aspects as explained by Runeson and Höst [18]; construct validity, internal validity, external validity and reliability.

1) *Construct validity*

Construct validity refers the degree of which the actual research conforms to its intention. The biggest threat to this study is ambiguity in the presented terminology. Since it is a framework, the parts need to be well defined. Further all test cases are assumed to have the same execution time and cost. This could impose a threat for the validity regarding practicability in industry.

2) *Internal validity*

Internal validity refers to the accuracy of the interpretation of the results, that no unknown factor is affecting the result. In the process of developing the framework, a tool implementing the framework is created. The results are depending on the correct implementation. Extensive verification on the tool is conducted to decrease this risk. However erroneous results from specific evaluations should not invalidate the framework as such.

3) *External validity*

External validity refers to the generalization of the findings, outside the studied context. The framework in itself is aimed to be general. But it is thinkable that there could be scenarios that are best handled outside the extent of the framework. Also the data used in this research might not be representative, but that would more impose a threat to single evaluations than to the framework itself.

4) *Reliability*

Reliability concerns the repeatability of the study. The research in this study can be hard to replicate due to the nature of the creative parts included in the design-science research. It is also hard to administer evaluation in design-science research [21, 22]. However given the proposed framework the evaluations should be possible to replicate.

E. *Alternative approaches*

The identification of relevant regression testing scenarios was conducted through discussions with two of the employees with good insight in the regression testing at the Ericsson site. However structured or semi-structured interviews with different stakeholders would have been an option. Since this thesis was carried out during the summer this approach was rejected because of summer vacations.

Concerning the collected data, there were more projects to gather information from. The number of projects used was regarded not to influence the structure of the framework itself but rather how good a RTS technique would perform. However, since the framework is based on a statistical approach, the amount of data within a project was of key value. It was decided to use only the project with the largest amount of information available.

IV. Framework foundations

In this section the parts of the framework is presented. There are seven building blocks presented, each representing a central part in the organization or the operation of the framework. Data models are used to represent data in different situations (i.e. analysis, selection, evaluation). Data preprocessing is used to manipulate data so that it is fit for analysis, selection or evaluation. Data analysis is used to gather more information about the collected regression test data and the testing environment. Scenarios state what the selection shall comply to. For instance a selection is to be made and a requirement could be that the selected test suite shall be 40% of the original size. The selection must then be made so that it adheres to this requirement. Scenarios also provide criteria which the selection techniques are evaluated on. In other words what the selection techniques try to optimize. Test case selections are the techniques used to select test cases for execution. Metrics are the measures used to present the evaluation of a selection given a specific scenario while evaluations present in what way RTS techniques are evaluated.

A. *Data models*

The data models are a central part of the framework and they are all presented as matrices. Each row represents a test case, each column represents a session and each cell contains a status value. When using historical test data four different data models appear. These models are presented in this section of the paper. Fig. 3 shows the relationships between the models.

1) *Raw status data*

The raw status data is collected directly from the testing environment. Depending on the storage of this data, data rearranging might be necessary, e.g. pivoting of test cases, replacing the notation of the test case status data, etcetera. The data will have different properties depending on the regression testing approach used, for instance if an RTS technique is already implemented the data will have a high amount of unexecuted test cases.

2) *Status data*

Preprocessing raw status data creates the status data presented in this paragraph. When conducting RTS evaluations continuously with the collected data the chosen selection methods will change the status data by setting the unselected test cases to null. For example if a selection is done on the first session, a number of test cases are selected and others are not. When making the next selection on the second session the test case status data for the unselected test cases from the first session will be set to null. These test cases are regarded as not executed when the selection for the second session is about to be conducted.

The question is now; what to do with failing test cases that are "nulled" by a selection? Two approaches are proposed. The first is to do nothing and the second is to let the "nulled" fail be exposed the next time the corresponding test case is executed. However this exposure is only valid given two assumptions. The first assumption is that the test environment is controlled and there are no non-deterministic or time-dependent faults. The second is that each fault correction only fixes one failing test case. This means that a correction of another visible failing test case would not fix the "nullified" fail. Since this "nullified" fail is not corrected it will be shown next time the corresponding test case is selected.

This is referred to as fail forwarding. Other approaches are possible, for instance, a statistical approach or solutions supported by a more extensive data analysis.

3) *Selection data*

The selection of test cases gives the selection data where the selected and unselected test cases for each session are presented as a matrix in the same manner as the raw status and status data. The cells contain either S for selected test cases, U for those not selected and X for non existing test cases.

4) *Evaluation data*

RTS evaluation uses the evaluation data which is created the same way as the status data but without having the unselected test cases set to null for the current session. The reason for this is that all available information is needed for a proper evaluation. But if fail forwarding is used then the forwarded fails must be used in both the evaluation- and the status data.

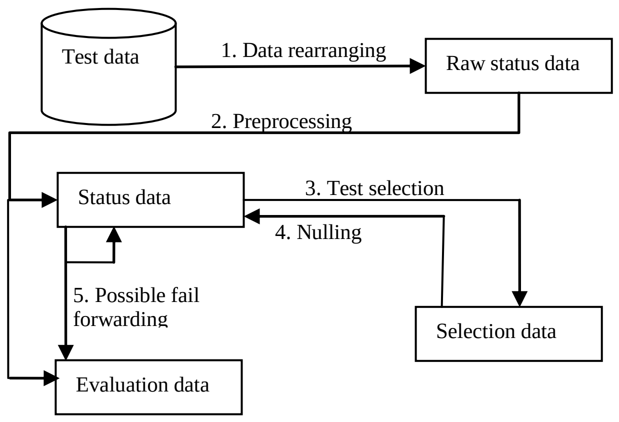


Figure 3. Overview of data models.

B. *Data preprocessing*

In order to conduct proper analyses and evaluations, preprocessing can be necessary. The data preprocessing can be divided into two parts; filtering and patching.

Filtering is about limiting or removing situations which have an unwanted effect on the analysis. For instance, test cases that are not properly executed for a single session might not be of interest and can therefore be removed.

When evaluating RTS techniques it is required to replace unknown status data; namely test cases with null values. This is referred to as patching. To accomplish patching an assumption about the "true" status of the test case needs to be made. Two basic approaches are presented, the 'non-faulty' where an omitted test case is replaced by a pass status and the 'most-recent' where the status from the previously executed session for that test case is chosen as replacement. These procedures are explained in [19] (however in a different context).

C. *Data analysis*

The purpose of data analysis is to collect more information about the data and the test environment. Basic information of interest could be fail density, null density, executed test cases per session, etcetera. There is an abundance of different information that can be extracted and this is just an excerpt.

The analysis can also function as a basis for the creation of an RTS technique. One approach could be to analyze the similarity between the test cases regarding their executions and then divide them over several sessions putting similar test cases in different runs.

D. *Industrial scenarios*

Industrial scenarios present realistic situations practitioners could face when dealing with regression testing. Scenarios state what requirements the selection technique must fulfill and they also present the approach for the evaluation as well as the chosen metric.

A scenario can be described in the following way:

1. A requirement that the selection must follow, for instance a test suite reduction of a certain percentage.
2. What criterion the evaluation shall be based on and what type of metric is to be used.

It is important to note that the choice of RTS technique is completely independent of the scenario. The scenario only states what the selection shall do, not how it should be done.

E. *Test case selection*

The entire test suite may, for every session, be divided into executed and non-executed test cases. One explanation for the non-execution might be that the test case is manually removed. Another reason could be that another RTS has been applied earlier and that it has not selected this test case. Yet another reason might be that the test execution have halted, leaving test cases un-executed.

When applying an RTS technique test cases are selected from the entire test suite (see Fig. 4) and therefore the selection is expected to include both executed and non-executed test cases. Then assumptions need to be made about the non-executed test cases (both the selected and the unselected) in order to perform a proper evaluation. This is referred to, in earlier sections, as patching.

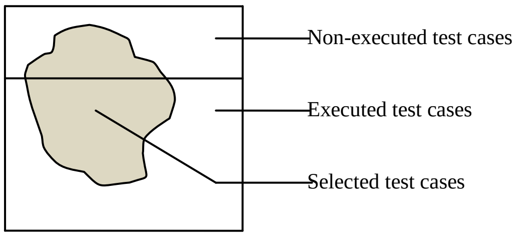


Figure 4. Test case categorization.

In terms of choosing an RTS technique this is, as stated, not dependent on the given scenario. Any selection technique could be used whether it is based on code changes, historical test data or if it derives from the knowledge of the developers.

F. *Metrics*

Every evaluation needs a metric. In research there are a couple of different metrics used and, as stated earlier, it is the design of the study that determines the choice of metric. In this study however, metrics are chosen based on what evaluation criterion the selection should optimize. Since an evaluation adapts to a given scenario, the metric also needs to reflect this.

The starting point, for analyzing regression test data and evaluating the selections, is metrics. Firstly there are atomic metrics, for instance the number of fails for- and the size of a given set of data. These can be combined into other base metrics like for instance Fail Detection Ratio (FDR) and Fail Detection Efficiency (FDE).

Base metrics are used to analyze test data, create RTS techniques and to evaluate them. Some base metrics, such as Fault Detection Efficiency [2] are used in previous research when evaluating RTS techniques. In Table I an excerpt of both common atomic and common base metrics is presented.

TABLE I. EXAMPLES OF ATOMIC AND BASE METRICS

Name	Description
Fail Count	The number of failing test cases for a given set of data.
Size	The size of a given set of data.
Fail Detection Ratio (FDR)	The number of detected fails for a selection divided by the total number of fails.
Fail Detection Efficiency (FDE)	The number of detected failing test cases for a selection divided by the size of the selection.
Test Case Selection Ratio (TCSR)	The size of a selection divided by the total size of the given set of data.
Fail Omission Quotient (FOQ)	The number of undetected failing test cases divided by the size of the unselected test cases.

G. *Evaluations*

Evaluations can be divided into single-session and multi-session evaluations. Single-session evaluations only assess one execution of test cases while the multi-session equivalent evaluates selections made on several consecutive sessions. Base metrics are primarily used for single-session evaluations.

Multi-session evaluations differ a bit from single-session evaluations. The reason for evaluating consecutive executions derives from the idea that it might not be crucial to execute all test cases every time. Instead an evaluation can indicate whether a selection technique has good or bad coverage over multiple sessions. The challenge here is to make the multi-session evaluation metrics dependable and understandable.

Also connected to the evaluations is the concept of sliding evaluation which is used to analyze both single-session and multi-session evaluation over time. It may also be used for aggregate measures of the historical test data, for instance average or mean.

V. COMPARATIVE STUDY OF RTS TECHNIQUES

In order to evaluate the proposed framework a detailed, comparative case study was conducted on two different RTS techniques for a given scenario. This case study describes a possible workflow when using the framework. To give further substance to the evaluation, additional comparisons were conducted and summarized. Two more scenarios were explored for four RTS techniques together with a best-case and a worst-case selection. These scenarios were identified in collaboration with Ericsson and each of them is presented with parameters regarded as relevant in the given context. Table II presents the scenarios.

TABLE II. CASE STUDY SCENARIOS

IS1	Test Suite Reduction (TSR40)
	Decrease the size of the test suite with 40%. What percentage of the failing test cases is selected?
IS2	Fail Omission Risk (FOR20/20)
	It is acceptable to have a 20% risk to miss 20% of the failing test cases. How large a percentage of the test cases would be selected?
IS3	Session Division (SD3)
	Decrease the size of the test suite with one third. Which test cases can be regarded as failed when examining three sessions?

A. *Case study*

The workflow for the conducted case study is shown in Fig. 5 below. It is suggested that this process is used in the proposed framework.

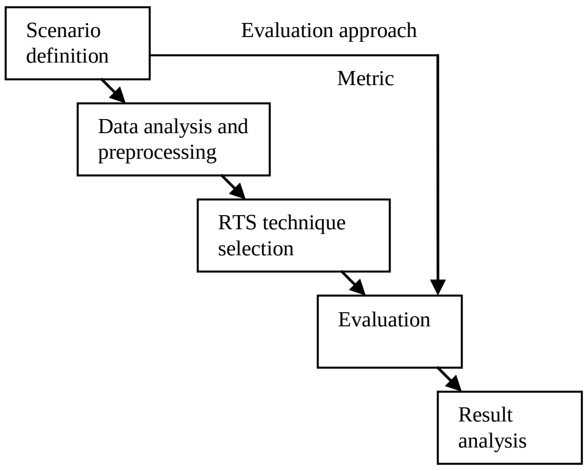


Figure 5. General workflow

The first step is to define a scenario which specifies the objective the RTS techniques, is to reach or optimize. This definition also stands as the basis for the choice of metric, and later in the process the evaluation approach.

After the scenario definition, the next step is to analyze the collected data and process it for further use. An analysis commonly conducted is the distribution of the number of fails each test case detects. This gives an indication of whether there are a few test cases which stand for the majority of the fails or if the fails are more evenly distributed.

Preprocessing could be the removal of test cases that have not been executed or the replacement of null values.

The next step is to choose the RTS technique or techniques to be evaluated and then conduct the actual test case selection. When that is completed the evaluation takes place. The selected scenario sets the stage for the evaluation and from that an analysis of the result is performed.

1) Scenario definition

In this comparative study the requirement given by the scenario was to decrease the amount of executed test cases by 40% (TSR40). The evaluation criterion was the percentage of failing test cases detected.

2) Data analysis and preprocessing

The data from the system referred to as release A was used. This data consisted of almost 400 sessions and there were nearly 4100 distinct test cases. Analysis of the data showed that there were four test cases that were null or not applicable for an entire session. Those test cases were thus removed. The data now had a null-status density of 3%. The next step was to make an assumption about the null-valued test cases. The most-recent patching approach was chosen.

3) RTS technique selection

The first RTS technique chosen was introduced by Fazlalizadeh et al. [20]. The technique will be referred to as Faz and was selected since it is based on the use of historical test data, which was the data available for this study. Also, the technique has been used in other studies [2].

Faz is originally presented as a prioritizing technique where a test case is given high priority by a combination of historical fault detection effectiveness, execution history and the priority of the previous session. Each of these factors is weighted by different parameters as can be seen in (1).

$$PR_k = \alpha * \frac{f_{ck}}{e_{ck}} + \beta * PR_{k-1} + \gamma * h_k \qquad (1)$$

$0 \leq \alpha, \beta, \gamma \leq 1, k \geq 1$

In this study an adaptation had to be made since the initial priority, as proposed by Fazlalizadeh et al., is based on code coverage. This was not the case for this study; instead, no initial prioritizing was performed, treating all test cases as equals. The original technique also dealt with faults occurring in each test case. In this study no such information was available so one failing test case was regarded as one fault. Furthermore the parameters were all set to 0.3, so no tweaking of the technique was made.

Since a sliding evaluation was undertaken, the assumption was made that fails “hidden” by the unselected test cases became visible the next time these test cases were executed.

The second technique used in this comparative study was random selection, referred to as Ran.

4) Evaluation

The evaluation was based on a base metric; in this case Fault Detection Ratio (FDR) i.e. the number of failing test cases detected divided by the number of total fails in the session. The metric was chosen based on the stated evaluation criterion given by the scenario. A sliding single-session evaluation with FDR was performed in all sessions and an average of the single-session evaluation value was calculated. This was performed on 60% of the test cases for each session, both for Faz and Ran. In addition to this selection, sizes of 10%, 20%, and so on, up to 90% were evaluated for further analysis.

5) Results

Based on the evaluation results a graph showing the average value of FDR for the different test selection sizes was constructed. Both the Faz- and the Ran technique is included (see Fig. 6).

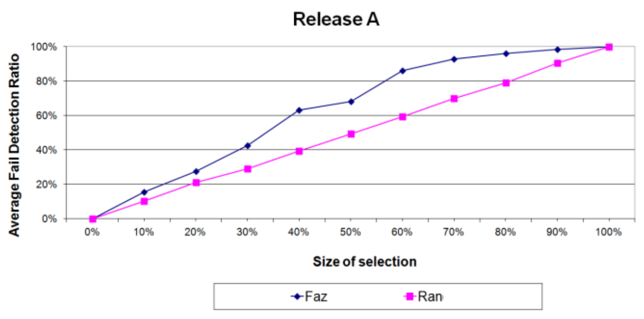


Figure 6. Average fail detection ratio depending on size for Faz and Ran.

First the results showed that Faz gave a higher average fail detection ratio for almost all test selection sizes. The biggest difference in fail detection ratio between the two RTS techniques became noticeable where the size of the selection was 60% of the original test suite. Furthermore, each increase of the selection size gave a relatively higher increase of the average FDR (except for 50%) up until a size around 60%.

The distribution of the FDR in all sessions for Faz and Ran respectively are plotted in Fig. 7. It shows that the probability of at least a 40% fail detection ratio is high for both Faz and Ran when the selection size is around 60%. When pursuing a fail detection ratio of more than 50% Faz was superior to Ran, as Faz had almost a 50% probability of finding at least 90% of the failing test cases.

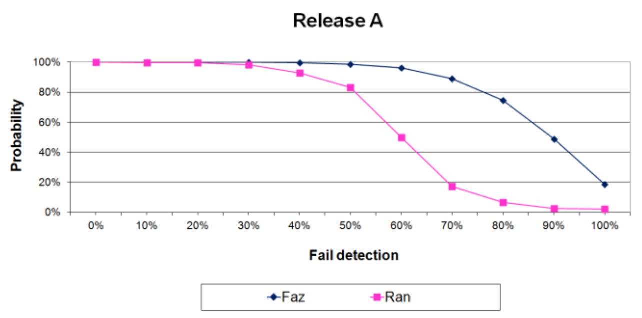


Figure 7. Probability for Faz and Ran to find at least a certain percentage of fails when selecting 60% of the test cases.

B. *Summary of comparison of RTS techniques*

The scenarios that were specified in section IV was used for the comparative study. The techniques used were Faz and Ran together with a slightly tweaked version of Faz. Also included was a selection scheme which divided the execution of the test cases over three sessions. This scheme was referred to as DivRan3. For comparative measures a best-case and worst-case test case selection was included. These were referred to as Best and Worst.

Yet again the data from release A was used. The same configurations as in the comparative study were assumed, namely a removal of test cases that were null or not applicable on all sessions. The 'null' test cases were replaced with the most recent test case status.

When it came to the tweaked version of Faz, it was lightly based on analyses of the data. Since only a few test cases were failing emphasis was made on the historical fault detection effectiveness factor. The parameters were set to 0.9 for α, 0.05 for β, and 0.1 for γ. This approached is referred to as Faz_tweak.

Scenario TSR40 was, as described in the previous sub-section, using FDR as base metric. It was also using a sliding evaluation in order to calculate an average FDR. The evaluation was plotted with average FDR over the selection size together with the fail detection distribution for a size reduction of 40%.

The second scenario used TCSR as base metric. In order to calculate this value several evaluations were conducted for different selection sizes with FDR as the base metric. For each selection size the risk of missing the given percentage of failing test cases was taken from the FDR distribution. A slight increase of the test selection size was used for each evaluation until the risk value closest above the stated acceptable risk was found.

In the third scenario, the base metric used was FDR; but, it was defined differently. The evaluation was of multi-session character and detection was noted when at least one failed test case was selected in one of the, in this case three, sessions. It was regarded an undetected failure if there was a failing test case in one of the sessions and no detection of it was made. This metric used for this scenario is referred to as FDR-bin3. The value of the FDR-bin3 was calculated by dividing the number of failing test cases detected with the sum of both detected and undetected failing test cases. The evaluation of scenario three was a sliding evaluation using all values from the collected data.

The results of the evaluations are presented in Table III which shows that the Faz_tweak technique presents the best value for all three scenarios.

TABLE III. COMPARISON OF RTS TECHNIQUES

RTS	Scenario		
	TSR40	FOR(20/20)	SD3
	Avg. FDR	TCSR	Avg. FDR-bin3
Faz	0.86	~0.68	0.86
Ran	0.59	~0.86	0.64
Faz_tweak	0.94	~0.52	0.87
DivRan3	N/A	N/A	0.87
Best	1	~0.01	1
Worst	0	~0,99	0

VI. DISCUSSION

The expensive nature of regression testing forces companies to search for more cost effective solutions. In case the regression testing is a bottle-neck for development one solution could be to acquire more hardware and in addition test cases could be independently created. Consequently allowing the test cases to run in parallel and thus decreasing the execution time. However, this could introduce problems of a pragmatic nature, such as where to put the hardware, as well as how to provide electricity and sufficient cooling. Buying more hardware is not sustainable in the long run and could cause environmental as well as image concerns.

In research there is a great range of proposed solutions regarding RTS techniques but the focus has been on RTS techniques which demand information about the connections between code and test cases. Such information is not always available and the maintenance of such information could be a problem in itself.

RTS techniques are most often only compared to a reference method such as retest-all or random selection [6, 15] which does not provide sufficient material to compare techniques. Furthermore, several studies are using small systems which might invalidate the generality of the results. There is a lack of large-scale evaluation in real situations as well as thorough comparisons of different RTS techniques. Engström et al. state [15] that it is hard for practitioners to make decisions based on research since most existing techniques are not sufficiently evaluated.

Studies conducted on evaluation of RTS techniques in industrial context (RQ1) are rare. The few that has been carried out have mainly focused on firewall approaches [7, 8, 9, 10]. In the four studies a total of six different RTS techniques were evaluated and one technique, the high-level firewall approach, appeared in three of them.

Metrics used for evaluation are either based merely on the structure of the conducted study [15] or on what is used as a de-facto standard in research. This paper suggests that the metrics used should be based on the criterion that test selections are evaluated on. This is the justification behind the definition of base metrics in this study.

The proposed framework is based on a statistical analysis (RQ3) and a comparative study shows that evaluations can be made independent of the technique used for selection (RQ2). Also, after evaluating the framework, examples were constructed on how RTS can be compared and evaluated (RQ4) which would imply that practitioners can profit using this framework. Besides the comparisons of RTS techniques there are possibilities for practitioners to increase their knowledge about the testing environment through analysis of the historical test data.

The case study had a two-sided objective; evaluating the framework and comparing RTS techniques. Insight in how a historical-based RTS technique, in form of Faz, performs was presented. In Table III it is shown that more than 90 % of the failing test cases are detected on average when the test suite is reduced with 40 %. This is achieved using only historical test data.

With easily accessible regression test data a quite high FDR is achieved and the question at hand here is how much more it will cost in order to get those last percentages of the FDR.

There are limitations to this framework. It is not able to categorize safe and unsafe RTS techniques since it is based on a statistical taxonomy and not one that is code-based. However the framework can give indications whether a technique shows such a behavior. With that said, extra measures would be needed for safety-critical systems.

The framework requires only the tests in the test suite without a connection to specifications or requirements. This means that the issue whether a test suite has a good coverage or not is not supported by the framework.

One problem is when an RTS technique has already been implemented in a real operation; the amount of information for each test case will then decrease. If a new RTS technique is to be evaluated, it demands that well educated assumptions about the missing information are made. An introduction of a risk measurement regarding the missing information would be a necessity. The possibility to evaluate RTS techniques on just the executed data (disregarding the non-executed) is one possibility, but whether the result from such an evaluation is adaptable to the whole data set is unclear.

Since the framework is based on statistics the amount of available data is of importance. However, this would not imply a limitation when using the framework; it just emphasizes the importance of continuous collection of data.

VII. Conclusion and further work

This paper was intended to review what empirical studies had been conducted on how to evaluate RTS techniques with an industrial perspective, to explore if such techniques could be objectively evaluated, to examine how a framework could be created to support this evaluation and to study how a categorization could be made based on the effectiveness of the selection approach.

Based on statistics, the proposed framework and its implementation open a possibility to analyze regression test data along with evaluation and analysis of different RTS techniques. Through comparative studies coupled with different realistic scenarios the framework was evaluated. In addition, the framework decreased the gap between academia and research in the sense that RTS techniques can now be evaluated based on information easily obtainable in an industrial context. Given the results it is now possible to state that one RTS technique is better than another in certain situations given specific criteria. This provides software developing organizations with a cost-effective and practical way to improve their regression testing. During the evaluation of the framework it was showed that the RTS technique proposed by Fazlalizadeh et al. [20] gave better average fail detection ratio for any size of the selected test cases compared to a random selection. When analyzing the data and tweaking the parameters given in Faz an even better average fail detection ratio was achieved.

The approach to divide the test cases over a couple of sessions has its justification. However, how to measure this conclusively is still not clear. The presence of non-deterministic or time-dependent faults is not supported in the framework and can impose a limitation to the test case division approach.

Regarding future research more information can be added as input data to the framework. For instance introducing the cost for each test case improves the framework and makes it more applicable. The relevance for industrial use is of utmost importance so a qualitative study over which regression testing scenarios are relevant could improve the validity of the framework.

Further studies on how test data analysis correlates to different regression testing situations with respect to granularity, test case dependency, etcetera, could be initiated and statistical models could be created. With a successful study in this area, practitioners could be aided in their choice of regression testing technique. Also the exploration of test executions profiles could give correlative behavior among test cases and be the basis of a division scheme.

ACKNOWLEDGMENT

The author is thankful to Associate Professor Robert Feldt at the Dept. of Software Engineering, Chalmers Technical University and Dr. Perolof Bengtsson and Bertil Aspernäs at Ericsson in Karlskrona for their support.

REFERENCES

[1] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: an empirical study," in *Proceedings of the International Conference on Software Maintenance*, Sep. 1999, pp. 179-188.

[2] E. Engström, P. Runeson, and A. Ljung, "Improving regression testing transparency and efficiency with history-based prioritization – an industrial case study," *in Proceedings of the 4th International Conference on Software Testing, Verification and Validation*, Mar. 2011, pp. 367-376.

[3] E. Engström and P. Runeson, "A qualitative survey of regression testing practices," in *Proceedings of the 11th International Conference on Product Focused Software Development and Process Improvement (PROFES '10),* June 2010, pp. 3-16.

[4] H. K. N. Leung and L. White, "Insights into regression testing," in *Proceedings of the Conference on Software Maintenance*, Oct. 1989, pp. 60-69.

[5] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability, vol. 21,* no. n/a, pp. n/a, Apr. 2010.

[6] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information and Software Technology*, vol. 52, no. 1, pp. 14-30, Jan. 2010.

[7] A. Orso, N. Shi, and M. J. Harrold, "Scaling Regression Testing to Large software Systems," in *Proceedings of the 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Nov. 2004, pp. 241-251.

[8] M. Skoglund and P. Runeson, "A Case Study of The Class Firewall Regression Test Selection Technique on a Large Scale Distributed Software System," in *2005 International Symposium on Empirical Software Engineering*, Dec. 2005, pp. 74-83.

[9] L. White, K. Jaber, and B. Robinson, "Utilizations of Extended Firewall for Object-Oriented Regression Testing," in *Proceedings of the 21st IEEE International Conference on Software Maintenance*, Nov. 2005, pp. 695-698.

[10] L. White and B. Robinson, "Industrial Real-Time Regression Testing and Analysis Using Firewalls," in *Proceedings of the 20th IEEE International Conference on Software Maintenance*, Nov. 2004, pp. 18-27

[11] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," *IEEE Transactions on Software Engineering*, vol. 22, no. 8, pp. 529-552, Aug. 1996.

[12] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 929-948, Oct. 2001.

[13] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, "Cost-cognizant test case prioritization," Technical Report TR-UNL-CSE-2006-0004, Mar. 2006.

[14] B. Qu, C. Nie, B. Xu, and X. Zhang, "Test case prioritization for black box testing," *in proceedings of the 31st Annual International Computer Software and Applications Conference*, vol. 1, Jul 2007,, pp. 465-474.

[15] E. Engström, M. Skoglund, and P. Runeson, "Empirical evaluations of regression test selection techniques: a systematic review," in Proceedings *Second ACM-IEEE international symposium on Empirical software engineering and measurement,* Oct. 2008, pp. 22-31.

[16] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information system research," *MIS Quarterly*, vol. 28, no. 1, pp. 75-105, Mar. 2004.

[17] W. Kuechler and V. Vaishnavi, "Design [science] research in information systems," Jan. 20, 2004, last updated Aug. 16, 2009, [Online]. Available: http://desrist.org/design-research-in-information-systems [Accessed: June 15 2011]. (Also published as work in progress: in *Proceedings of 2nd International Conference on Design Science Research in Information Systems and Technology (DESRIST '07),* vol. n/a, no. n/a, pp. n/a, May 2007.)

[18] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164, 2009.

[19] M. Rashid, "Evaluation of regression test effectiveness," MS Thesis, Chalmers University of Technology, Göteborg, Mar. 2011, final draft, unpublished.

[20] Y. Fazlalizadeh, A. Khalilian, M. Azgomi, and S. Parsa, "Prioritizing test cases for resource constraint environments using historical test case performance data," *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 190-195, Aug 2009.

[21] M. V. Zelkowitz and D. R. Wallace, "Experimental models for validating technology," *Computer*, vol. 31, no. 5, pp. 23-31, May 1998.

[22] W. F. Tichy, "Should computer scientist experiment more?," *Computer*, vol. 31, no. 5, pp. 32-40, May 1998.

[23] T. L. Graves, M. J. Harrold, J.-M. Kim, A. Porter, and G. Rothermel, "An empirical study of regression test selection techniques," *Transactions on Software Engineering and Methodology (TOSEM),* vol. 10, no. 2, pp. 184-208, Apr. 2001.