# A Value-preserving Approach to Regression Test Selection in Agile Methods

by

Aniruddha Mondal

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Computer Science

Approved November 2023 by the
Graduate Supervisory Committee:

Kevin Gary, Co-Chair
Srividya Bansal, Co-Chair
Ayca Tuzmen

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

This thesis introduces a requirement-based regression test selection approach in an agile development context. Regression testing is critical in ensuring software quality but demands substantial time and resources. The rise of agile methodologies emphasizes the need for swift, iterative software delivery, requiring efficient regression testing. Although executing all existing test cases is the most thorough approach, it becomes impractical and resource-intensive for large real-world projects. Regression test selection emerges as a solution to this challenge, focusing on identifying a subset of test cases that efficiently uncover potential faults due to changes in the existing code. Existing literature on regression test selection in agile settings presents strategies that may only partially embrace agile characteristics. This research proposes a regression test selection method by utilizing data from user stories—agile's equivalent of requirements—and the associated business value spanning successive releases to pinpoint regression test cases. Given that value is a chief metric in agile, and testing—particularly regression testing—is often viewed more as value preservation than creation, the approach in this thesis demonstrates that integrating user stories and business value can lead to notable advancements in agile regression testing efficiency.

ACKNOWLEDGMENTS

First, I would like to sincerely thank my thesis advisor Dr. Kevin Gary for keeping me motivated throughout my research tenure through constant guidance and support. He not only guided me through regular technical discussions and meetings but also cultivated a sense of ownership and innovation by sharing several valuable ideas. The various aspects of statistical analysis and data querying would not have been possible without Dr. Gary's expertise and the numerous hours he put in to augment my work.

I would like to express my sincere appreciation to my thesis co-chair – Dr. Srividya Bansal and committee member Dr. Ayca Tuzmen, for taking their valuable time out and kindly agreeing to attend my thesis defense. Thank you for taking an interest in my research.

With much love and gratitude, I want to thank my family and friends for providing constant support and encouragement throughout this journey.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

# CHAPTER 1

# INTRODUCTION

This thesis proposes an approach to select a subset of test cases for regression testing using requirement prioritization in agile environments. Software applications persistently evolve throughout their lifetime, adapting to various needs, such as rectifying issues or incorporating new functionalities, often driven by market and organizational requirements. Upon implementing these changes, it becomes necessary to ascertain the correct performance of the modified segments alongside ensuring the correctness of the already verified and delivered functionalities [1]. This verification exercise, termed regression testing, is instrumental in guaranteeing the system's overall quality [2]. The importance of software testing has grown with the great adoption of agile development methodologies. Agile is a methodology for product delivery, where changes occur rapidly which has brought testing in general, and regression testing in particular, much more into the center of software development [3].

Given the frequent changes of user requirements in an agile environment, executing all existing test cases—a straightforward strategy in regression testing can be time-consuming and can have detrimental effect on product delivery timelines. To address this problem, existing studies on regression testing consider various techniques in order to increase the efficacy of regression testing. One of such techniques is called regression test selection (RTS) [4–7] where a subset of test cases is selected from the original test suite. Test suite minimization (TSM) is another strategy that removes redundant test cases permanently from the available test suite [8, 9]. Whereas RTS and TSM both work on reducing the size of the original test suite, test case prioritization (TCP) technique assigns

execution priority to the existing test suite [2, 3, 10–15]. This thesis focuses on RTS, used to reduce the size of the test suite for regression testing by including the test cases alone that test the existing functionalities affected by the newly introduced changes [10]. This helps us to build a smaller regression test suite while ensuring the regression testing quality. Regression test selection procedure improves the overall efficiency of regression testing by reducing the cost, effort, and time of execution. Various factors have been considered throughout the literature that can impact the efficiency and effectiveness of RTS, such as fault detection history of test cases [8, 13, 14], code coverage [8], and user requirements [16-19]. Some studies have considered time constraints agile methods, and the emphasis on delivering business value as factors for their prioritization and selection approaches. For example, authors in [13, 14, 20] considered the variation in execution time, [8, 21] used the cost of each test case as a factor in their RTS approaches, and [3] used business value, a crucial agile attribute associated with the user stories, in their prioritization and selection approach. Previous works presented in [16-19, 22, 23] have proved that utilizing requirement information can improve the regression testing efficiency. For instance, a study by Chittimalli and Harold [18] depicts a significant reduction (86%) of test cases using requirement based RTS while ensuring high-quality testing.

A user requirement, demonstrating the intended behavior of a system, not only can be an essential factor in improving regression testing, but it is also an integral part of a software system's development as it serves as the basis for building the system. Thus, including the importance of requirements during the testing phase has been well-understood by the requirements engineering community [24]. However, some of these requirements are of greater significance—either due to higher user engagement or being

more susceptible to errors—than others [19]. Commonly, interconnected requirements are implemented within the same set of classes belonging to the same subsystem to maintain a software product's cohesion. Consequently, test cases tied to a set of similar or interconnected requirements tend to engage a similar set of classes [19]. In Agile context the requirements are typically expressed as user stories. For each new feature introduced, there is an associated user story. All user stories are specific to releases, where the feature described in the user story gets introduced. Regression testing ensures that introduction of these new developments or changes don't disrupt existing features, thus focusing on user stories from prior releases where these features originated. This signifies that for regression testing only the prior release user stories are looked at and test cases pertaining to these user stories are considered.

However, user stories also have an essential attribute called business value, representing their significance to the stakeholders, customers, or the organization. According to [9], the primary output metric in agile is the value delivered to the customer, measured by the business values associated with the user stories delivered. User stories add value through every new feature added and delivered with the required functionalities. Regression testing ensures all the existing functionalities remain intact for every new modification introduced, preserving the value already delivered. Business value of a user story is considered to be preserved if the selected regression test cases entirely test the functionalities of the delivered user story. This thesis utilizes this preserved value to compute the effectiveness of the proposed RTS approaches. The computation of the preserved value is performed based on the test cases selected and business values of their associated user stories. If all the test cases pertaining to a particular user story are selected

through the proposed RTS approach, the business value associated with this user story is considered to be preserved.

The studies [16, 18, 22, 23] showcase ordering requirements to enhance the effectiveness of RTS and RTP. Additionally, findings from [19] depict the application of textual similarity to order the requirements. However, all the proposed approaches try to measure the effectiveness based on the fault detection ability of the regression suite.

Motivated by the importance of value delivery nature of agile in conjunction to the success of requirement prioritization using textual similarity in regression testing, this thesis proposes an RTS approach that utilizes requirement prioritization. The requirement prioritization in this thesis is attained using two factors:

1. Textual similarity value between current and prior release user stories—ensures the most relevant prior release user stories get prioritized.
2. Business value of user stories—user stories most valuable to the business get prioritized.

Initially, textual similarity analysis is employed to identify the already delivered user stories affected by the new user stories. Subsequently, a requirement prioritization of the user stories from prior release(s) is devised based on their similarity to the new user stories and the associated business value. By creating a subset of test cases from this prioritization, the regression testing process becomes more focused on areas of the software where the potential for disruption is higher, ensuring adequate preservation of valuable existing functionalities. Additionally, in agile methodologies, the lack of substantial consideration for business value when assessing the effectiveness of regression testing prompted us to

explore an approach that uses value preservation as a metric for evaluating the effectiveness of RTS.

The primary contribution of this work is to select test cases for regression testing, associated with the existing user stories which deliver high value to the customers. This thesis also advocates the use of a crucial agile attribute to measure the effectiveness of the selected test cases. In this work we utilized augmented real-world user story data and experimented with the impact of context variables to evaluate the effectiveness of multiple variations of the value preserving approach on RTS in agile software environments.

The rest of the thesis is divided into 6 chapters. Chapter 2 focuses on literature review related to this research. This section has been divided into two subsections where the first subsection talks about the studies related to regression test selection and the second subsection gives an overview of the studies which incorporated regression test selection into agile software development. Research questions and research methodology used in this study are described in two different subsections of Chapter 3 delving into the overview of the processes involved. Chapter 4 focuses on the detailing of the implementation steps that have been followed in this thesis. This chapter comprises of two subsections talking about the factors that have been varied to generate the data and how the experimental setup has been done to run the simulations. Chapter 5 summarizes the observations of results obtained from the simulation runs. This chapter also focuses on queries that emerge from the findings of our study's visualizations. A further drill down study has been conducted on these queries and around the research questions in Chapter 6. The characteristics of this study are summarized in the last chapter, along with the future scope of this research.

# CHAPTER 2

# RELATED WORKS

Regression testing constitutes a requisite, yet resource-intensive maintenance endeavor conducted on modified software. Its primary purpose is to instill confidence when new code is added over the existing verified code [15]. While the default regression testing approach involves rerunning all such tests, this comprehensive retest-all strategy may incur excessive time and resources. In contrast, identifying test cases judiciously and running them in an efficient pattern for regression testing to test the specific requirements affected by the new changes may decrease this demand for time and resources to a greater extent.

Throughout the literature, strategies have been discussed trying to optimize regression testing. The studies [13, 25-30] considered for the literature review in this thesis discuss three major strategies in the context of regression testing to optimize the testing process in terms of time, resources, and cost namely Test Suite Minimization (TSM), Regression Test Selection (RTS), and Regression Test Prioritization (RTP) [31]. TSM is used to reduce the size of a test suite by removing redundant test cases without affecting the fault detection capability of the test suite. RTS on the other hand is the process of selecting a subset from the existing test suite by identifying the tests that are relevant only to test the changed parts of the software. Thus, both TSM and RTS focus on reducing the size of the test suite for regression testing. However, RTP orders the test cases based on some specific criteria such as the likelihood of finding defects, the importance of the functionality being tested, etc. This ensures that the most important or relevant tests are run first as the idea of RTP is to detect potential regressions as early as possible in the testing process which can be especially useful when there is insufficient time to run the entire test

suite [3]. RTP is considered important as “high priority defects” may be fixed early, before having to change delivery schedules. However, RTP is performed on a specific set of selected test cases which limits the total amount of delivered value that can be preserved by regression testing even when there is enough execution time. This thesis focuses on maximizing the possibility of preserving the already delivered value with the increase in the regression execution window. Thus, the thesis focuses on RTS, selecting test cases that focuses on existing user stories delivering high value to the customers.

In this thesis, we focus on the specificity of RTS, assuming that the set of all regression test cases is already minimized using an appropriate test suite minimization technique. As a result, the subsequent analysis focuses on this minimized set, ensuring that redundant tests are excluded.

The following subsections highlight the works on various approaches to improve regression testing in non-agile and agile software development setups.

### 2.1. Regression Test Selection (RTS):

Regression test selection is an established field of research and seminal papers such as [1] have presented approaches to tackle this problem. Below we present the works of relevant studies for this thesis that served as motivation.

**2.1.1. Code Based RTS:** Various studies on regression testing techniques have relied on code coverage information and many empirical studies have shown that these techniques can improve the effectiveness of regression testing [32-37]. A seminal paper by Rothermel and Harrold in 1997 introduced an RTS technique centered on program dependency analysis and test suite reduction [15]. The RTS techniques

proposed in this study first identifies program entities that are affected by the newly introduced modifications and construct a dependence graph. A heuristic algorithm selects a subset of test cases to cover the modified entities and their dependencies, aiming to maximize entity coverage while reducing the test case count. The technique proposed is primarily focused on detecting changes made to source code. Two major RTS algorithms have been proposed in the paper: Intraprocedural and Interprocedural. The intraprocedural test selection algorithm operates on individual procedures, whereas the interprocedural test selection algorithm operates on entire programs or subsections. The study evaluates the proposed RTS on six real-world software systems and compares their approaches. The results show that the saving in terms of time obtained from intraprocedural is not significant; on the other hand, interprocedural shows more significant time savings when applied to larger programs. The results in this study highlight that performing RTS for larger projects with a large volume of test cases is more beneficial than when applied to smaller projects with a small volume of test cases, as the analyzing time required to perform the test selection process involves extra cost in terms of time and resource.

**2.1.2. History-based RTS:** Throughout the literature, we came across other factors such as execution history of test cases [7, 8, 13, 38] which is being considered for RTS.

A history-based RTS technique, ReTEST [13], to reduce regression testing time and cost employs a history-centric approach, factoring in software changes since the last test run. This approach utilizes information retrieval techniques that extract relevant textual information from source code and other software artifacts (e.g., code commits, requirement documents, etc.). The authors used program

changes, test suite source files, and fault history information to identify a list of important test cases that are highly likely to detect regression defects. The proposed technique in this paper uses 3 major inputs: 1) term or textual similarity to the modified portion of the program, 2) test failure history, and 3) test diversity. To evaluate their approach an empirical study was performed on four open-source applications written in two different programming languages and compared its performance with three existing techniques, namely Safe Technique [4], Code Coverage Technique (tests selected using a greedy algorithm), and Test Dissimilarity (measured using the Jaccard Index). The results showed that ReTEST outperformed the existing techniques in terms of cost and time while maintaining a high level of fault detection. The Safe technique was found to be the most expensive approach of all the others as it tends to execute all the test cases available. Compared to the other two approaches ReTEST showcased a good amount of cost reductions.

**2.1.3. Requirement Based RTS:** A significant number of studies [16, 17 ,18, 19] address requirements-based regression testing. One of the studies that motivated this thesis is the technique proposed by Chittimalli and Harold [18]. The study provides a way to use system requirements and their associations with test cases to perform RTS. In addition, the proposed technique uses this criticality to order the selected test cases. The main benefit of this approach is that it provides a way to select a subset of test cases for use on the changed software that can instill confidence in the changes made. Another aspect of this technique is the use of only the association (coverage) of requirements by the test cases instead of requiring model or code

coverage, which is often not practical to gather. The experiments in the study used two real-world systems and was compared with the safe technique which selects all test cases for the regression suite. The results show that there is a significant percentage (86%) reduction of the test cases to be rerun to test the modified requirements. The results also show that the approach selects a few unnecessary test cases and compares well with the code-based approach that requires code coverage. Also, this experiment produced no false negatives, which means that, for the subject, requirements, test cases, and changes, the technique omitted no test cases that could show a difference from the original to the modified version of the software, and thus, omitting them was safe.

Another study by Srivastva et al. [16] proposed a requirements-based RTP technique which explores two factors: 1) the priority of requirements given by customers, developers, and managers, and 2) the severity and probability of risk factors that occur in requirements. The study calculates the risk exposure for each requirement, based on the assigned value of the severity impact by the developers. Based on the value of the risk exposure a weighted priority factor is created which is used to prioritize the requirements. A Priority Decision Table is created that assists testers in determining the sequence in which test cases should be executed. One of the major similarities our thesis holds with this study is calculating the importance of the requirements instead of the test cases.

Khalid and Qamar [23], collect a dataset of black box test cases and assign them user priority weights (according to the priority assigned during the requirement elicitation Phase) [39, 40] as in the simple ranking method [41]. The

data set has been sorted based on weights, and the K-means algorithm is applied to divide it into the "K" number of clusters. The approach calculates a priority percentage against each function point of test cases according to the requirements. K-Medoids [14] is applied based on calculated cost and time priority percentage on each split priority cluster, further dividing the data set into three more refined clusters. The results show 79.174% of the test cases according to business requirements are correctly prioritized as compared to the original prioritized data.

**2.1.4. Use of Textual Similarity in RTS:** To delve into finding similarities between requirements we explored papers capturing textual similarity and their usage in requirement-based regression testing.

For this thesis, the studies on requirement clustering and measuring textual similarity between requirements are the closest to this work and thus served a major motivation. Our research bears a close resemblance to the study presented in [19] by Arafeen and Do, which addresses requirements clustering for test case prioritization. To perform clustering the authors first use textual similarity to find resemblance amongst requirements and then use k-means to cluster the similar requirements together. These requirement clusters are then prioritized constructing a scale of weights based on two factors: 1) Code commits performed by developers, and 2) the amount of code modification. Requirement-test cases traceability matrix is then used to pick test cases from the prioritized set of requirements. Average Percentage of Fault Detection (APFD) [22] has been used to validate the results which show improvement rates ranging from 55.52% to 65.37%. This technique used code modification as a factor along with requirement similarity that demands

access to the source code which sometimes can be a complex task for large applications.

After thoroughly examining the above papers, the potential of the text similarity approach in advancing our research seems promising. This methodology aligns well with the objectives outlined in our thesis. Drawing from the findings of these papers, the text similarity strategy stands out as a critical mechanism to achieve our intended outcomes. Recognizing these pivotal studies is crucial as we delve deeper into the subsequent sections of this thesis.

### 2.2. Agile Regression Testing:

With the increasing adoption of agile development methodologies, the significance of quick software testing in an iterative manner has been magnified. Agile emphasizes swift product delivery with frequent modifications to gather faster feedback. Consequently, there is a swift transition from traditional testing methods like the waterfall model [42–44]. This shift is influenced mainly by evolving end-user expectations and the fluidity of development in response to product alterations. Regression testing, which ensures that new changes do not disrupt existing software functions, constitutes a notable portion of software development costs. Agile methodologies prioritize continuous integration thereby necessitating frequent regression testing in a time-boxed environment [45].

Kandil et al. [3] proposes an approach to prioritize and select test cases for regression testing in agile software development. The authors argue that traditional test case prioritization techniques are not well suited for agile development because they are time-consuming and require much information about the system. Instead, they propose a

clustering approach that groups similar test cases and selects a representative subset from each cluster for testing. The results show that the technique was more effective than random selection in terms of fault detection rate and testing efficiency. The authors also discuss the limitations of their approach, including the need for appropriate clustering algorithms and the potential for bias in the selection of representative test cases.

Ali et al. [46] introduced an augmented regression testing method suitable for agile software development and continuous integration paradigms. Their methodology leverages code coverage analysis and change impact analysis to select and prioritize test cases. Similarly, [47] evaluated various test selection strategies based on code coverage and dependencies. They emphasized that these strategies can markedly reduce the time needed for feedback without compromising the robustness of the test process.

Multiple studies [5, 48] present approaches of RTS within agile methodologies, emphasizing the CI/CD process. Elbaum et al. [49] discusses challenges and solutions for optimizing regression testing in continuous integration (CI) settings. The authors underscore that while CI promotes software quality and speed, it presents unique hurdles for regression testing. After evaluation on a large-scale project, the results effectively reduced regression testing time and effort which led to cost-effectiveness improvements in the continuous integration process.

Shi et al. [50] explores module and class level RTS methodologies within a CI context. Although module level RTS offers minimal overhead for analyzing module impacts, it can result in the selection of more test cases than class-level RTS. On the other hand, while class-level RTS can be more selective, it presents challenges, including potential omissions of affected tests and extended analysis durations. Utilizing Travis CI,

a leading cloud-based CI service, the authors assessed the performance of these techniques on a diverse range of open-source Java projects. The study collects three metrics relative to running all test cases for regression testing: the percentage of tests selected, the percentage of time to run the selected tests, and the percentage of time overall to build the jobs. The results indicate that the code-based RTS techniques do save testing time compared to running all tests (RetestAll), but the percentage of time for a full build using RTS (76.0%) is not low, due to the extra overhead in a cloud-based CI environment.

Another study by Yu and Wang [51] states that integrating many RTS techniques into CI is challenging. The authors suggest a comprehensive exploration of open-source projects to measure the necessity of RTS in CI environments. The rate of changes in open-source projects can determine the RTS strategy. Infrequent changes allow for traditional or broad dependency analysis techniques due to the luxury of time. At the same time, frequent alterations demand efficient RTS methods to reduce tests and analytical time. In this research, the authors assess 7,018,512 commits across 918 open-source projects employing CI. The goal is to determine the extent to which RTS is indispensable. The study investigates commit frequencies to discern how rapid alterations might influence RTS efficiency. The paper also evaluates the proportion of modified source files in commits to pinpoint which files merit attention. The results show that for 60% of commits, the time between them exceeds 10 minutes, indicating that if testing completes within 10 minutes, RTS might be redundant for these commits. Most changes in rapid commits (those within 10-minute intervals) focus on a mere 8.6% of source files, suggesting RTS can target these essential files when changes are swift. In 62.2% of Java projects, average testing durations are less than a quarter of average commit intervals, hinting at reduced RTS necessity. For

the majority (97.3%) of Java projects, method-level RTS is less efficient than testing all cases. However, class-level RTS proves more time-efficient than exhaustive testing for 56.8% of Java projects.

The domain of regression testing, both within and outside agile methodologies, has seen significant strides in methodological development and optimization. From techniques embedded in program dependency and test suite reduction to history-based approaches and clustering methods, there is a significant effort to maximize efficiency and fault detection. Central to these advancements are methodologies that emphasize code coverage, change impact analysis, and even the integration of Machine Learning. As software development methods evolve, especially with the rise of continuous integration, the summarized research highlights the criticality of an ongoing exploration in improving regression testing techniques.

The above studies have shown remarkable ways in which the efficiency of regression testing can be improved. Many of these studies have incorporated their approaches in an agile environment. Some of the crucial agile attributes were also seen to have been used in many of the studies. Despite incorporating agile attributes as factors in the selection or prioritization approach, the effectiveness of the techniques was measured by their capacity to find either more severe faults or faults faster. Though these metrics are helpful, this thesis suggests the involvement of a true agile attribute, the business value associated with the user stories, in measuring the effectiveness of the selection process. Also, taking into account the time framed nature of agile, this thesis considers the effect of varying execution window, i.e., time available for executing the regression suite. The following section talks about the research question and the methods followed in this thesis.

# CHAPTER 3

# RESEARCH METHODOLOGY

The focus of this thesis is to ensure that the existing user stories delivering high value to the customers are tested in regression testing when a new change is introduced. This aim demands the selection of all the test cases associated with the existing user stories delivering high value with an objective to understand the effect of prioritizing requirements based on their value delivering ability and the available time for executing the regression suite on RTS approaches. To achieve this goal this chapter focuses on some specific research questions mentioned in the following subsection.

## 3.1. Research Questions

*RQ1. How does the inclusion of user story business values influence the selection of regression tests in the context of agile software development?*

*RQ1.1 How does the use of business value from current release user stories affect test case selection in terms of value preservation?*

*RQ1.2 How does the use of business value from prior release user stories affect test case selection in terms of value preservation?*

*RQ1.3 How does the combination of business values from prior and current release user stories affect test case selection in terms of value preservation?*

*RQ2. How does the duration of test case execution influence the selection of test cases during regression test selection?*

*RQ2.1 How does the selection process in regression test selection change when all test cases exhibit equal execution times?*

*RQ2.2 How does the value preservation in regression test selection change when regression execution window is varied?*

To answer these research questions, this thesis employs a large-scale simulation by augmenting a real-world established dataset, implements a custom requirements-based regression test selection process, and evaluates performance of that process by measuring the value preserved. The following subsection gives an overview of the steps involved in this method.

### 3.2. Methodology

This thesis follows the method of prioritizing the existing user stories that have been impacted by the new changes based on the value they are currently delivering. This prioritized list of user stories is then used to select the test cases associated with them to ensure the complete testing of the high value delivering user stories and preserve the maximum possible delivered value. Figure 1 summarizes our approach's main activities and how these activities are related to each other.

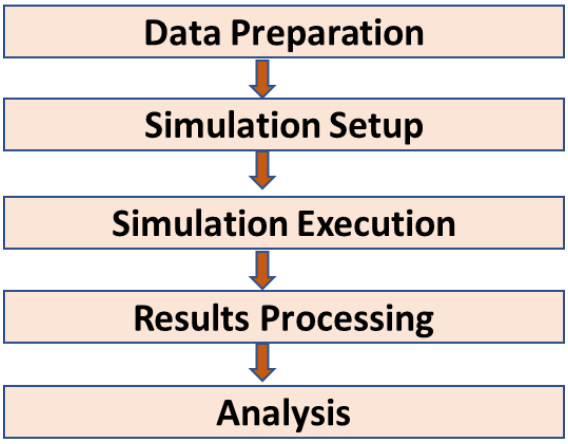


Figure 1: Summary of the activities followed in this thesis.

The following subsections describe the activities mentioned in Figure 1 in detail.

### 3.2.1. Data Preparation

To run the simulations for this thesis the data needed to have a few specific characteristics: 1) User stories written in natural language, 2) User stories had to be categorized into current and prior releases, 3) User stories had to have business values assigned to them, and 4) Test cases had to be assigned to the user stories. Multiple studies [38, 52-55] were investigated to identify the availability of relevant data for this research. Though these studies had a good collection of usable data the focus of this thesis was to obtain user specifications or user stories written in natural language. The user stories were found in the required format in a well-maintained condition in a specific dataset: the TAWOS dataset [53], which have been used as the starting point for the purpose of this thesis. As mentioned above the need for the remaining three parameters for the data was also crucial. TAWOS dataset provided the user stories in the required format, but the remaining three parameters were present with limited availability. For running the simulations there was also a need to vary the above parameters to replicate multiple real-world projects. Due to this limited availability the user stories written in natural language were extracted from the TAWOS dataset and the remaining parameters were synthesized for the purpose of its usage in this thesis. The synthetization of the remaining specifications also ensured that the variability of the data could be maintained to simulate multiple real-world agile projects. For this step of the process, the extracted user stories were categorized into two releases: New (Current) and Existing (Prior), and the user story–test case traceability matrix was created. A user story-test case traceability matrix is a mapping to

identify the test cases associated with each user story. Figure 2 is the pictorial representation of the processes.

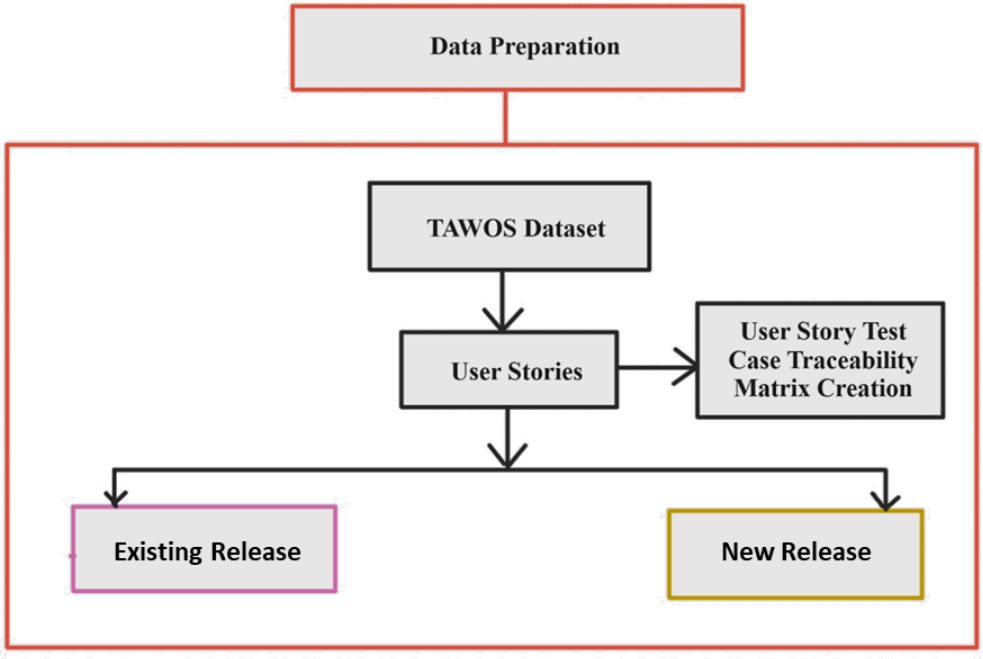


Figure 2: Processes involved in data preparation.

### 3.2.2. Simulation Setup

Once the data was prepared the datasets were created to ensure the variability of entities across all datasets. These datasets were used to run the simulations. The factors that were varied for each dataset are: 1) Execution time available in regression testing, 2) Business values of each user story in any given dataset, and 3) Count of user stories in either release. Fibonacci series numbers were used as business value for the user stories. Varying all these factors, separate datasets were created against which the proposed approach was run. Figure 3 shows the processes involved in this step.

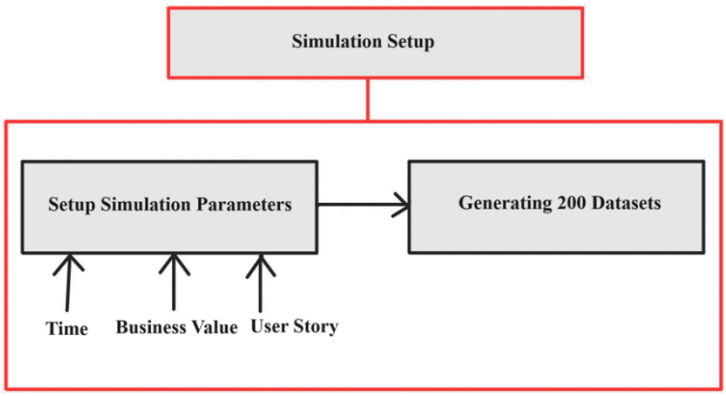


Figure 3: Processes involved in the simulation setup.

### 3.2.3. Simulation Execution

After the datasets were prepared and the data for the simulation run was ready the simulation was run against all the datasets created. The textual information of the user stories from both releases was picked and compared against each other to identify the impact of the new release user stories on the existing release user stories through a textual similarity comparator. The textual similarity value was combined with the business value of the user stories to calculate an importance-value. This importance value is a keyword used specific to this thesis to prioritize the user stories. The computed importance-values were associated with the existing release user stories. Once the importance value was associated with each user story, they were ordered in decreasing order of their magnitude which resulted in the prioritized list of the user stories. This prioritized list of user stories and the user story-test case traceability matrix was then used to start the test case selection process. The selection of test cases starts from the user story having the highest importance-value. This provides the required list of selected test cases. Figure 4 shows the processes involved in this simulation execution step.

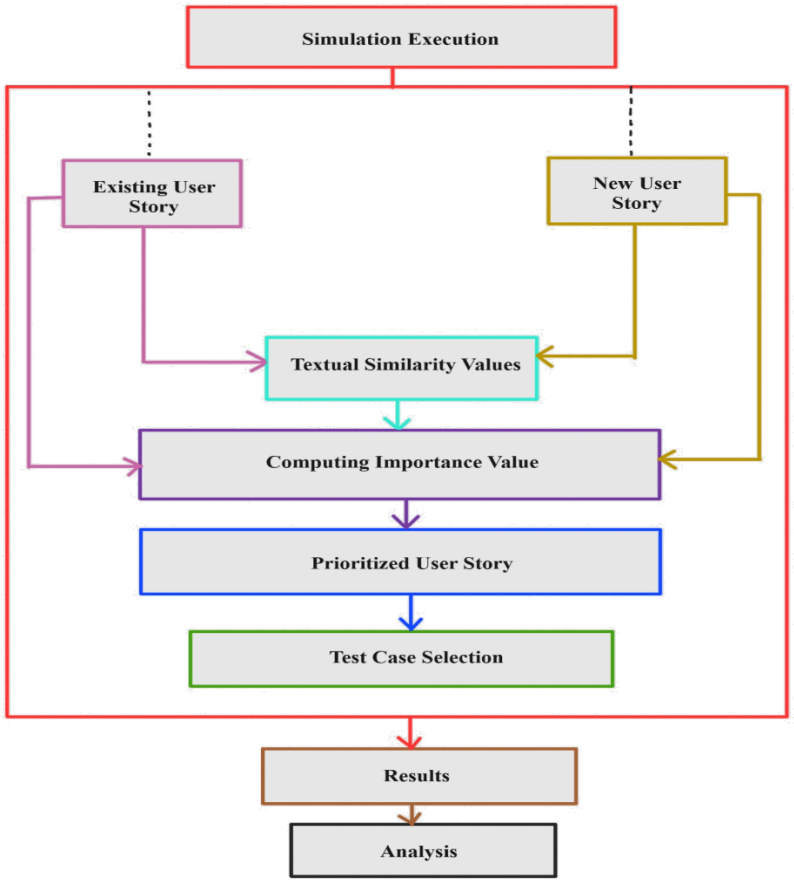


Figure 4: Processes involved in simulation execution, results, and analysis.

### 3.2.4. Results

After the simulations had been run for all the datasets the results were collected. The results were stored in the database and in an Excel file parallelly. The database was used to store the results for future use. The results after exporting to Excel were used to visualize the data using various plots and bar charts and compare them against the baseline approaches. The visualization of the data was used to better analyze the results.

### 3.2.5. Analysis

On generating the results, descriptive statistics were computed for each set of data for all datasets to analyze them. The results were looked at individually as well as in groups

to analyze them in depth. The visualizations were used to understand and analyze the results better and find any kind of similarity in their characteristics. The analysis was done to find the effectiveness of the proposed RTS approach in comparison to the baseline approaches and answer the research questions mentioned at the start of chapter 3.

The above processes are an overview of the methods followed to examine the approach proposed in this thesis. Requirement prioritization has been widely used showcasing improvements in the effectiveness of RTS. With the improvement in language models the use of textual similarity has shown promising results in finding similarities and connections between texts. Also, incorporating a true agile attribute, the business values of user stories would portray the effect of inclusion of this approach in an agile environment. These factors due to their effectiveness and strong reliability worked as strong motivation in incorporating them for the approach in this thesis.

The following implementation chapter provides the details of all the steps mentioned in this methodology section. It also provides a step by step procedure for performing the required simulation to be able to answer the designated research questions.

# CHAPTER 4

# IMPLEMENTATION

The proposed test selection process is a multi-factor selection process that explores two factors while selecting a subset of test cases: (1) textual similarities between the current release and prior release user stories; (2) the business value associated with the user stories. This section describes the implementation of the proposed RTS technique taking the above-mentioned factors into consideration. The following subsection highlights the input factors and their variability. Section 4.1 focuses on these factors which have been varied to generate the datasets for this thesis.

## 4.1. Factors Influencing Data Variation in Dataset Generation

This thesis focuses on multiple key factors to simulate real-world agile projects for developing regression test selection algorithms. These factors are crucial in reflecting agile software development's dynamic and complex nature. Following are each of the factors and the reason to simulate a real-world agile project.

### 4.1.1. Real-World Agile Projects:

1. Business value: In agile development, delivering business value is paramount [9]. Agile methodologies prioritize features and tasks based on their potential impact on the business or end-users. In the context of regression testing, understanding the business value of user stories ensure that testing efforts align with the most critical aspects of a project. This alignment is essential for simulating real-world agile projects, as it mirrors how decisions are made in actual development environments.

This thesis uses Fibonacci series numbers from 1 through 89 as business values to associate with the user stories as Fibonacci series numbers have been seen as a common proxy for business value numbers [66]. For all the datasets generated, user stories in half of the datasets have a uniform distribution of business values and for the other half, the distribution is right skewed, i.e., a greater number of user stories have lower business values. This gave an overall average of right-skewed distribution for all the datasets.

2. Count of user stories at release level: Agile projects are characterized by iterative development, each bringing new user stories. The count of user stories in various releases is an actual measure of this iterative process and the evolving nature of the software. This thesis captured agile environments' dynamic workload and scope by varying this count. This variability is crucial for regression test selection algorithms, reflecting the real-world challenge of adjusting test suites to continuously changing software features and requirements. To simplify the release level understanding this thesis categorized the user stories into two releases. One release is the new or current composed of user stories considered to have introduced new changes into the software system which brings in the need for regression testing. Another release is the existing or prior release which contains user stories that have been already delivered. Test cases associated with the prior release user stories are the candidates for the regression suite. This factor has been varied for every dataset with a uniform distribution.
3. Regression testing execution time/window: The time allocated for regression testing can vary significantly in real-world agile projects. Agile teams constantly

change software to meet customer needs, requiring efficient regression testing to prevent faults. The variation in regression testing time captures the real-world challenge of fitting comprehensive testing into tight agile schedules. This consideration is essential for developing regression test algorithms that are not only effective but also time efficient. Figure 5 shows the possible requirement of regression testing (RT) at various levels in an agile environment.

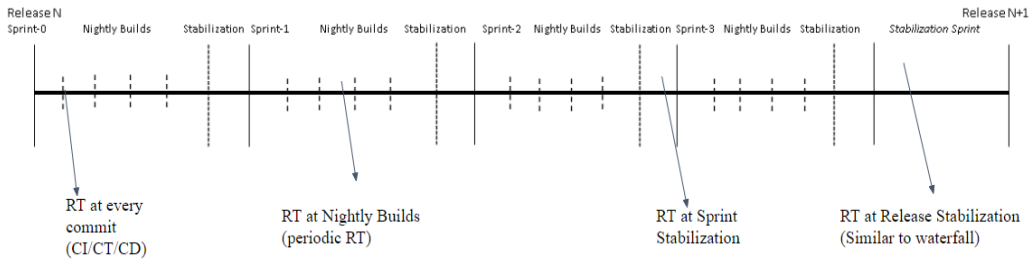


Figure 5. Regression testing execution windows in agile environment

The variation of time at different levels for performing regression testing fluctuates immensely. This thesis considers this variation and assigns a different execution window for each dataset. The distribution of this factor over all the generated dataset is also uniform.

The varied execution time of test cases reflects the complexity and dependencies in real-world software projects. It provides a time-based weightage for every test case which can be an important factor in addition to other factors when selecting test cases for regression testing. Acknowledging this diversity is vital for creating realistic test environments and optimizing test suites for both coverage and efficiency which has been recognized by [8, 20, 21]. In the scope of this thesis execution time of each test case has been considered as constant and no time-based weightage has been given to the test cases while performing the selection process. Another factor, test case history is critical to

regression testing in agile environments. It provides insights on previous failures, successes, and modifications of test cases. The test case history reports provide the ability to understand how effective a test case is depending on the number of sever defects it has found when previously executed. Though these test case level factors and many others can prove to be effective in making an informed decision while selecting the test cases, our thesis in the due interest of time does not consider these factors within its scope. This thesis focuses on incorporating some major factors, like the regression execution window, business value and count of user stories in releases to address the real-world agile aspects.

#### 4.1.2. Factors derived from the variations in input

Certain factors were derived from the varying inputs mentioned in 4.1.1 for each dataset to examine the results. These factors facilitated an exploration of how the results varied in conjunction with the fluctuations in various attributes across the dataset. The following sections presents a detailing on these derived factors:

1. **Number of selected test cases ($\#TC_{selected}$):**

$$\#TC_{selected} = \frac{Total\ execution\ window}{Execution\ time\ of\ a\ single\ test\ case} \quad (9)$$

   This factor provides the aggregated quantity of test cases feasible within the designated timeframe allocated for regression testing. The duration required for the execution of each test case remains constant across all instances in this research study.

2. **Percentage (%) of test cases selected ($P_{tc}$):** This metric represents the proportion of chosen test cases relative to the overall count of available test cases. The following formula is used to compute the metric:

$$P_{tc} = \left(\frac{Total\ number\ of\ test\ cases\ selected\ by\ RTS}{Total\ number\ of\ available\ test\ cases}\right) \times 100 \quad (5)$$

This factor is used to analyze the change in the business value preserved when the percentage of test cases selected is varied.

3. **Average business value of prior release user stories ($ABV_{US^{CR}}$):** This factor is used to study the average value of business value pertaining to each user story in a particular dataset, calculated by (6).

$$ABV_{US^{CR}} = \frac{Total\ business\ value}{Total\ number\ of\ user\ stories\ in\ previous\ release} \quad (6)$$

This factor is normalized to bring all the data to a standard range of 0 to 1. The datasets in this thesis comprise variables that originate from diverse scales and units of measurement. Without normalization, direct comparisons across these variables would be inherently flawed due to discrepancies in scale and magnitude. The below formula has been used to normalize all the values obtained from this factor.

$$Normalization(x) = \frac{(x - Minimum\ value)}{(Maximum\ value - Minimum\ value)} \quad (7)$$

where:

Minimum value = Minimum value in the range of values to be normalized;

Maximum value = Maximum value in the range of values to be normalized.

4. **Ratio of current release to previous release user story counts ($US_{CRPR}$):** This is computed using (8):

$$US_{CRPR} = \frac{Total\ number\ of\ user\ stories\ in\ the\ current\ release}{Total\ number\ of\ user\ stories\ in\ the\ previous\ release} \quad (8)$$

This metric serves as a tool for evaluating the structural intricacies of a project. Its primary purpose is to emulate a real-world project. When a notable increase in the overall count of releases and the cumulative number of user stories across preceding releases significantly surpasses that of an individual release, i.e., the ongoing release,

the metric tends to exhibit a lower value. Usually, this scenario is observed in the case of large projects and demands regression test selection due to large volume of test cases, where running all test cases is arduous, time-consuming, and costly.

Figure 6 is a scatter plot of the variation of the factors across all 200 datasets.

Figure 6: Scatter plot of all four factors across all 200 datasets

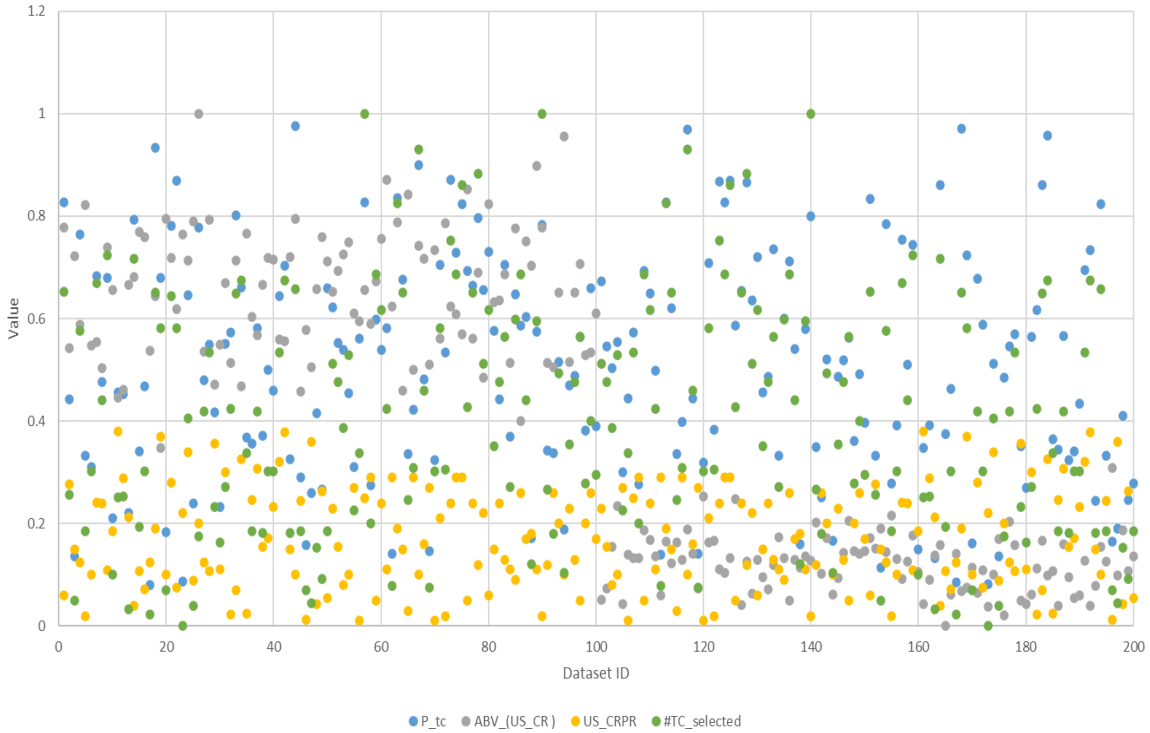


Table 1 gives the descriptive statistics of the four factors across the 200 datasets.

| | $P_{tc}$ | $ABV_{US^{CR}}$ | $US_{CRPR}$ | $\#TC_{selected}$ |
|---|---|---|---|---|
| **Mean** | 0.506254994 | 0.377853022 | 0.172704033 | 0.403334496 |
| **Median** | 0.507506973 | 0.33462491 | 0.165 | 0.399860433 |
| **Mode** | 0.826446281 | 0.16218593 | 0.1 | 0.302163294 |
| **Std Deviation** | 0.221609753 | 0.267395549 | 0.099415989 | 0.237087523 |
| **Variance** | 0.049110883 | 0.07150038 | 0.009883539 | 0.056210494 |
| **Skewness** | 0.036929363 | 0.183303084 | 0.176156019 | 0.316316442 |
| **Range** | 0.895082994 | 1 | 0.37 | 1 |
| **Minimum** | 0.080645161 | 0 | 0.01 | 0 |
| **Maximum** | 0.975728155 | 1 | 0.38 | 1 |

Table 1: Descriptive Statistics of the 4 factors across all 200 datasets

The provided data encompasses four distinct factors, each shedding light on various aspects of the dataset. $\boldsymbol{P_{tc}}$ indicates that, on average, approximately 50.63% of test cases are chosen for execution, with a modest standard deviation of 22.16%, suggesting moderate variability in test case selection. $\boldsymbol{ABV_{US^{CR}}}$ reflects the normalized business value associated with each user story, averaging around 37.79% and displaying a slightly positively skewed distribution (0.1833), signifying variations in the business value attributed to the individual prior release user stories.

$\boldsymbol{US_{CRPR}}$, quantifies the ratio of user stories between the current release and prior releases. On average, this ratio stands at approximately 17.27%, indicating the structure of large projects where the current release has very few user stories compared to all prior releases. Lastly, $\boldsymbol{\#TC_{selected}}$ examines the normalized value of test cases selected, averaging about 40.33% and exhibiting a positively skewed distribution (0.3163). This suggests that most datasets selected a relatively low number of test cases. These statistical factors collectively provide valuable insights into the dataset's central tendencies, variabilities, and distributions, enabling a comprehensive understanding of the dataset's characteristics and aiding in informed data-driven decisions.

These variations in the input factors are made to incorporate the different structural variations noticed in real-world projects. Our data is synthesized based on the initial extraction of the user stories from the TAWOS dataset. Due to the lack of all required relations with the data a way was devised to simulate the links we need within the data. As a large part of our data was generated it was important to try and capture all possible variations in the input factors to replicate the possible real-world projects. This was made

possible by varying the factors in this subsection. The following subsection speaks about the process and the setups performed in this thesis for running the simulations.

## 4.2. Experimental Setup

The variation in the factors mentioned in 4.1 was used to generate 200 datasets. 200 datasets were sufficient to cover a wide range of differences in the four factors. This wide variety is clearly shown in the scatter plot in Figure 5, highlighting the thoroughness of the data and its ability to capture different factor combinations. To simplify the process, the implementation of the RTS strategy is categorized into six primary activities: (4.2.1) calculating similarities between user stories; (4.2.2) mapping user stories to test cases; (4.2.3) computation of an importance-value for the prior release user stories; (4.2.4) process execution for this thesis; (4.2.5) selecting test cases based on the importance-value of the user stories; and (4.2.6) computing the efficacy of the selected test cases. The following subsections describe each activity in detail.

### 4.2.1. Calculating similarities between user stories

Inspired from the works on requirement prioritization [16, 18, 22, 23] and use of textual similarity [3] for clustering similar requirements, this thesis uses textual similarity comparison between two user stories to measure their resemblance. Textual similarity between two user stories suggests how much one user story relies on or is connected to the other. This is a measure of the impact the new user stories have on the existing ones. Thus, textual similarity is used to create the mapping between the user stories of the current release and the user stories of prior release(s) based on their interdependency. The categorization of the extracted user stories was done as mentioned in the “Data

Preparation" section of Chapter 3, described in the above subsection ($US_{CRPR}$) across all 200 datasets. Following this categorization, we implement the textual similarity between each of these user stories across the releases, specific to each dataset.

To find the textual similarity, Python library spaCy is utilized to identify the above-mentioned textual similarities. spaCy [56] is an NLP tool that helps us to find the textual likeness between texts. Existing research [57] has portrayed the use of various Python libraries to compare the similarities between texts. However, when it comes to comparing domain-specific texts, the choices are narrowed down. Our task was to choose an efficient tool that could provide us with satisfactory results when comparing texts specific to the software engineering domain. Out of the most widely used tools according to [58-60] NLTK, Stanford CoreNLP, Sentistrength were found to be less efficient in understanding software engineering texts. A direct comparative analysis of these tools with spaCy, as presented in [61], has demonstrated spaCy's superior performance. Notably, spaCy has exhibited enhanced capabilities in sentiment analysis within the domain of software engineering. Additionally, the study conducted in [62] has showcased spaCy's efficacy in extracting features from software requirement specification documents. The studies [63, 64] use spaCy to create conceptual models from user stories that state its implementation and efficiency. Also, it shows that the standard models of spaCy with little or no tuning can perform efficiently in context to their usage in the software engineering domain.

spaCy provides us with a cosine distance measure of similarity between -1 and 1, where -1 means most dissimilar, 0 means no similarity and 1 means complete similarity. In the context of this research, any value below 0 has been treated as 0 owing to the minimal influence of the similarity value. To find the similarity, each user story of the current

release is compared to each of the user stories of the previous release(s). The similarity value is then used in combination with the user story business values in step D to compute the importance-value for the prior release user stories.

### 4.2.2. Mapping of user stories to test cases

In executing any RTS process, establishing test cases is essential, as a subset of these cases represents the ultimate output of the RTS process. Every test case is mapped to a user story as they test the functionalities these user stories aim to deliver. A requirement-test case traceability matrix is a mapping between the user stories and their relevant test cases, used to track their relationships. Starting with the user stories obtained from TAWOS [53] we synthesize test cases and set it onto the database. Since this thesis does not have any use with the test case descriptions, test case ids have been created with the test case execution time. A total of 700 test cases have been created for the purpose of running the simulations. The number of test cases per dataset has been varied as per the variable "$Total\ number\ of\ available\ test\ cases$" in $P_{tc}$ mentioned in subsection 4.1. Once the input has been fed for the number of test cases and the number of user stories (as mentioned in A) to be used for any dataset the requirement-test case traceability matrix is created. We are considering requirement-based test case creation which brings us to a one-to-many user story – test case mapping. It was ensured that each test case is mapped to a single user story, but one user story can have multiple test cases mapped to it. This mapping has been done by creating an adjacency matrix using the Python programming language which has been created using the random function of the python library. Once the adjacency matrix is created the mappings of the test case and user stories are stored in a database table.

Following the creation of test case user story mapping, the importance-value is computed as below.

### 4.2.3. Computation of the importance-value

Importance-value specifically used in the context of this thesis is used to prioritize the user stories for the RTS approach. Since regression testing focuses on the prior release user stories, we aim to associate the computed importance-value to each of the prior release user story. This is a value that is computed taking the similarity value, and business value associated with the user stories into account. The importance-value is calculated to make an informed decision while selecting the subset of test cases. The consideration of the factors in calculating the importance-value plays an essential role in deciding the overall impact of a user story in creating the regression suite. The importance-value computed and associated with any prior release user story are mapped to the test cases associated with the particular user story. To address the research question RQ1 we are considering the business value of the user stories along with the similarity value to calculate the importance-value. The business values have been normalized to bring them to a common range which would help us to calculate the importance-value irrespective of their units. To normalize the values, the below approach (9) has been used.

- Let values = [value1, value2, value3] be a list of values.
- Normalizer = 1/(value1+value2+value3)
- N(value1) = [value1×Normalizer]

Where, N(value1) is the function to compute the normalized value for value1. (9)

This normalization process ensures that the values maintain the proportionality of the original values while making them collectively add up to 1. Using any other form of normalization would fetch similar results which has been tested by our approach. The following subsection presents the ways of computing the importance-value in the context of this thesis.

### 4.2.4. Process Execution

In regression testing the existing functionalities are tested, functionalities that have been built prior to the current release. This indicates the need for prioritizing the user stories of the prior release(s) and not the current, thus the importance-value is calculated and associated to all the prior release user stories. Following this, ordering of the user stories is carried out based on the computed importance-value which is the combination of these two factors: 1) textual similarity value, calculated between two user stories ($TSV(US_i^{CR}, US_m^{PR})$), one from the prior release ($US_m^{PR}$) and one from the current ($US_i^{CR}$), and 2) the business values associated with different user stories ($US.businessvalue$). This thesis advocates three different configurations for computing the importance-value based on different possible combinations of the business value from the current and prior release user stories. These configurations have been created to highlight the impact of the business value while prioritizing the user stories in the creation of the regression suite and analyzing their effectiveness. This thesis uses two more configurations as the baseline purpose: Random Selection Process and SimOnly. The Random selection process is an industrial standard process whereas SimOnly is a configuration that has been created specific to this thesis. SimOnly uses a single factor: textual similarity value, to compute the importance-value for

the existing user stories to analyze the effect of business value on the other RTS approaches.

Below are the details of the various configurations used in this thesis:

**Random Selection Process (RSP):** In the realm of software testing, practitioners frequently opt for the random selection of test cases or draw upon their existing knowledge and experience when there are no test selection tools available [65]. The random selection process is an example of such a technique that involves the random sampling of a specified percentage of test cases from the pool of all test cases**.** RSP is used as one of the baselines to compare and analyze the effectiveness of the configurations advocated by this thesis.

**SimOnly:** This configuration involves only one factor and does not consider any of the user story business values for the computation of the importance-value as shown in (10). It computes the importance-value solely based on the textual similarity value found between the user stories. The significance of SimOnly is to compare and understand the importance of considering the business values in the proposed RTS process. SimOnly is used as the second baseline for analyzing the results.

$$IV(US_m^{PR}) = \sum_{i=0}^{n} TSV(US_i^{CR}, US_m^{PR}) \quad (10)$$

where,

$US_i^{CR} = i^{th}$ user story of the current release;

$US_m^{PR} = m^{th}$ user story of previous release(s);

$IV(US_m^{PR})$ = Importance-value for $US_m^{PR}$;

$TSV(US_i^{CR}, US_m^{PR})$ = Similarity value when textual comparison is performed between $US_i^{CR}$ and $US_m^{PR}$.

On describing the two baseline approaches the following section gives a detailed description of the three advocated configurations. Table 2 gives an overview of the factors involved in the three configurations.

| Configurations | Factors involved | |
|---|---|---|
| | Factor1 | Factor2 |
| SimBiz_New | Textual Similarity Value | Business value of current release user stories |
| SimBiz_Existing | Textual Similarity Value | Business value of prior release user stories |
| SimBiz_Combined | Textual Similarity Value | Business value of both prior and current release user stories |

Table 2: Factors involved in SimBiz_New, SimBiz_Existing, SimBiz_Combined

**SimBiz_New:** In SimBiz_New, alongside the textual similarity the business values of current release user stories are also considered to compute the importance-value, based on (11). The business value associated with the current release user story ($US_i^{CR}.businessvalue$) has been given the equal weightage along with the textual similarity value $TSV(US_i^{CR}, US_m^{PR})$ for calculating the importance value using this configuration.

$$IV(US_m^{PR}) = \sum_{i=0}^{n} (TSV(US_i^{CR}, US_m^{PR}) \times US_i^{CR}.businessvalue) \quad (11)$$

where,

$US_i^{CR}.businessvalue$ = Normalized Business value of $US_i^{CR}$.

**SimBiz_Existing:** On contrary to SimBiz_New, in SimBiz_Existing, the business value of the previous release user story is considered for the computation of the importance-value. Both the textual similarity value and the business value of the prior release user story are given equal weightage for computing the importance value as shown in (12).

$$IV(US_m^{PR}) = \sum_{i=0}^{n} (TSV(US_i^{CR}, US_m^{PR}) \times US_m^{PR}.businessvalue) \quad (12)$$

where,

$US_m^{PR}.businessvalue$ = Normalized Business value of $US_m^{PR}$.

**SimBiz_Combined:** Lastly, in SimBiz_Combined, the business value of the user story from both the current and prior release(s) along with the textual similarity value are considered for the computation of the importance-value. All the three factors hold equal weightage when calculating the importance-value using SimBiz_Combined and gets associated with the prior release user story ($US_m^{PR}$) as shown in (13).

$$IV(US_m^{PR}) = \sum_{i=0}^{n} \left(TSV\left(US_i^{CR}, US_m^{PR}\right) \times US_i^{CR}.businessvalue \times US_m^{PR}.businessvalue\right) \quad (13)$$

The above-mentioned configurations were used to compute the importance value for every user story of the prior release. The computed importance-value is then used to prioritize the user stories with the highest importance-value having the highest priority. On having a prioritized list of user stories, the test case selection process starts as detailed below.

### 4.2.5. Selecting test cases based on the importance-value of the user stories:

After determining the importance-values for all the prior release user stories, the user stories are ordered in decreasing order of the associated importance-value. The selection process begins with selecting test cases associated with the highest prioritized user story. We consider the timeframe allocated for executing the regression suite, which is the total permissible time for regression execution, factor "$Total\ execution\ window$" from $\#TC_{selected}$. This timeframe is set against the individual execution times of each test

case "*Execution time of a single test case*" and the possible number of test cases that can be executed within the given timeframe is selected to build the final regression suite.

### 4.2.6. Computing the efficacy of the selected test cases:

The regression suite obtained from the RTS approach is evaluated to check if all the test cases associated with a particular user story are selected to measure the effectiveness of the approaches. To preserve the business value of a user story this work considers that all test cases associated to that user story must be selected through RTS. If all the test cases associated with a user story is present in the selected set of test cases through the RTS process, the business value of the user story is preserved as shown in Figure 7, any exception to this, the business value of the user story is not accounted as preserved. The figure depicts that all the test cases pertaining to $US_i$ are a subset of the selected test cases and in such a case the business value of $US_i$ is considered to be preserved.

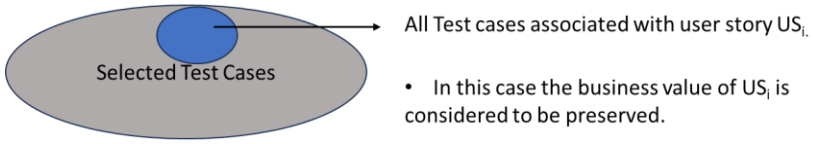


Figure 7: Computation of value preserved.

All the results are exported to an excel as well as the database. The results are stored to the database for future reference. The results exported to excel are used to compute the descriptive analysis on the data. Plots are drawn using the data to better visualize and compare to the baselines mentioned in "Computation of the importance-value" section of this chapter. The maximum possible value preservation is computed for every dataset which is the value of the business value that can be preserved if all prior release user stories

and their associated test cases are considered without taking into account the impact of current release user stories. The ratio of the value preserved is computed with respect to the maximum possible value preservation for every configuration separately for all the datasets which is crucial to understand the effectiveness of each configuration across all the datasets.

Following the steps mentioned in this section the simulations were run for 200 datasets and the results were fetched. The results were fetched into an Excel for all the datasets separately and were analyzed to answer the research questions. The following chapter depicts the results obtained from the simulation runs.

# CHAPTER 5

# RESULTS

In this section, we present the results for all the configurations discussed in the implementation section. To comprehend the results an analysis has been performed on the data obtained. Table 3 displays the descriptive statistics of the results of all the configurations for all 200 datasets.

| Descriptive Statistics of Results | | | | | | |
|---|---|---|---|---|---|---|
| | Configurations | | | | | |
| | **Maximum preservable value** | **RSP** | **SimOnly** | **SimBiz_New** | **SimBiz_Existing** | **SimBiz_Combined** |
| **Mean** | 1264.39 | 428.65 | 739.54 | 738.41 | 1218.36 | 1218.42 |
| **Median** | 897.5 | 278.5 | 571.5 | 581.5 | 863.5 | 861.5 |
| **Mode** | 686 | 86 | 1096 | 395 | 2513 | 2513 |
| **Standard Deviation** | 794.89 | 443.42 | 591.87 | 592.96 | 777.76 | 778.73 |
| **Range** | 3552 | 2154 | 2725 | 2731 | 3449 | 3447 |
| **Minimum** | 118 | 0 | 7 | 11 | 114 | 114 |
| **Maximum** | 3670 | 2154 | 2732 | 2742 | 3563 | 3561 |
| **Count** | 200 | 200 | 200 | 200 | 200 | 200 |

Table 3: Descriptive Statistics of all the configurations across 200 datasets.

All the configurations have been run across 200 datasets. In the interest of space, the results for 50 datasets are shown below. Remaining results are provided in Appendix B. The following subsections exhibit the results of the business value preserved by each configuration with respect to the maximum preservable business value.

## 5.1. Result for each configuration:

In this section the result for each configuration has been shown in comparison to the maximum preservable value.

### 5.1.1. Random Selection Process (RSP):

Random Selection Process as described in the implementation section selects a random set of test cases for the regression suite with performing any prior prioritization. Table 3 shows an average value preservation by RSP is 428.65 which is lowest of all the other configurations. Figure 8 shows the results for 50 datasets. From the figure it can be observed that for dataset 18 and 44 the value preservation is relatively high compared to other datasets. This can be reasoned by the high percentage of test cases selected for the two datasets, 93.452% and 97.57% respectively.

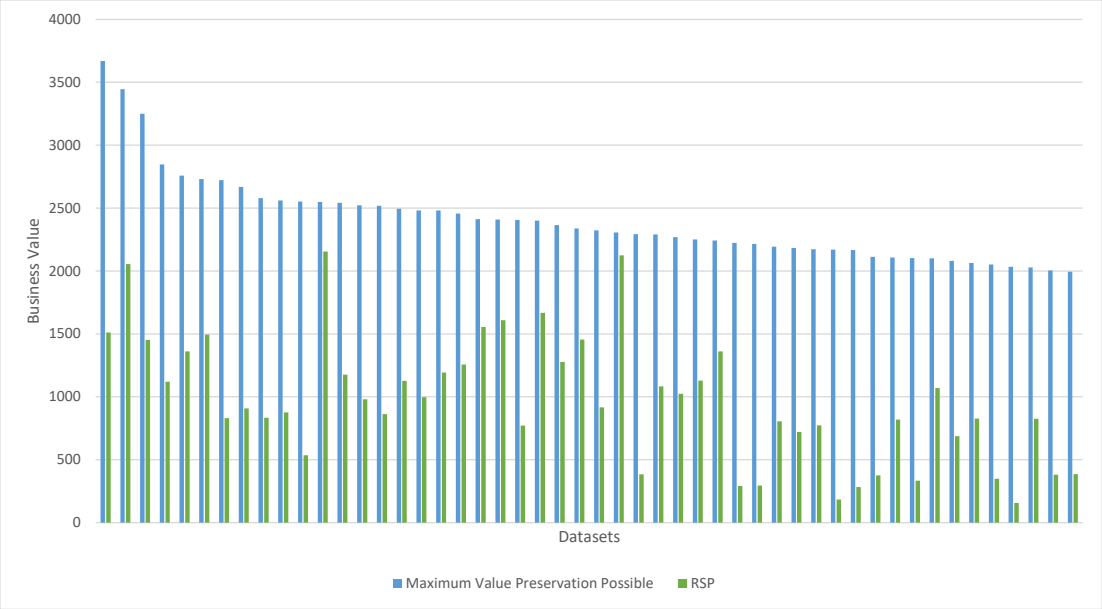


Figure 8. Maximum preservable value vs. RSP

### 5.1.2. SimOnly:

Figure 9 depicts the results for 50 datasets in comparison to the maximum preservable business value. SimOnly considers only the textual similarity value for the computation of the importance-value. The business values of user stories from either

release are not considered in this case. As shown in Table 3 the average value preservation by SimOnly is 739.54 which is very similar to SimBiz_New.

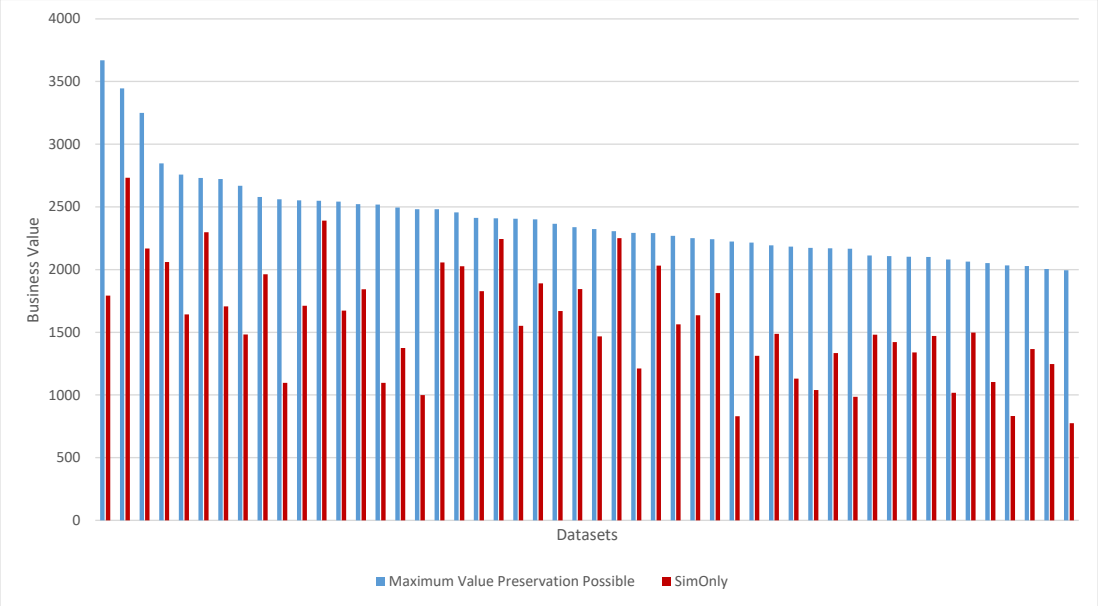


Figure 9. Maximum preservable value vs. SimOnly

### 5.1.3. SimBiz_New:

SimBiz_New involves textual similarity value and the business value of the current release user stories for the computation of the importance-value. From the data in Table 3 it can be observed that the average value preserved by SimBiz_New is 738.41 which is just above half the average of total business value measured for all 200 datasets. Figure 10 shows the bar chart of the business value preserved by SimBiz_New in comparison to the maximum preservable business value for 50 datasets.

### 5.1.4. SimBiz_Existing:

SimBiz_Existing considers the textual similarity value and the business value of the previous release user stories for the computation of the importance-value. Table 3 shows that the average value preserved by SimBiz_Existing is 1218.36 which is very

close to the average of total business value across the 200 datasets. SimBiz_Existing shows good value preservation when compared to other configurations. On average it preserves 96.35% of the maximum preservable business value. Figure 11 depicts the results of SimBiz_Existing for 50 datasets.

### 5.1.5. SimBiz_Combined:

SimBiz_Combined considers the textual similarity value, the business value of the previous release user stories, and the current release user stories to compute the importance-value. The statistics in Table 3 demonstrates the high performance of SimBiz_Combined where the average value preserved is 1218.42, much close to the average of maximum preservable value of 1264.39. It is also seen to have performed very similar to SimBiz_Existing. Figure 12 shows the value preserved by SimBiz_Combined for 50 datasets in comparison to the maximum preservable business value.

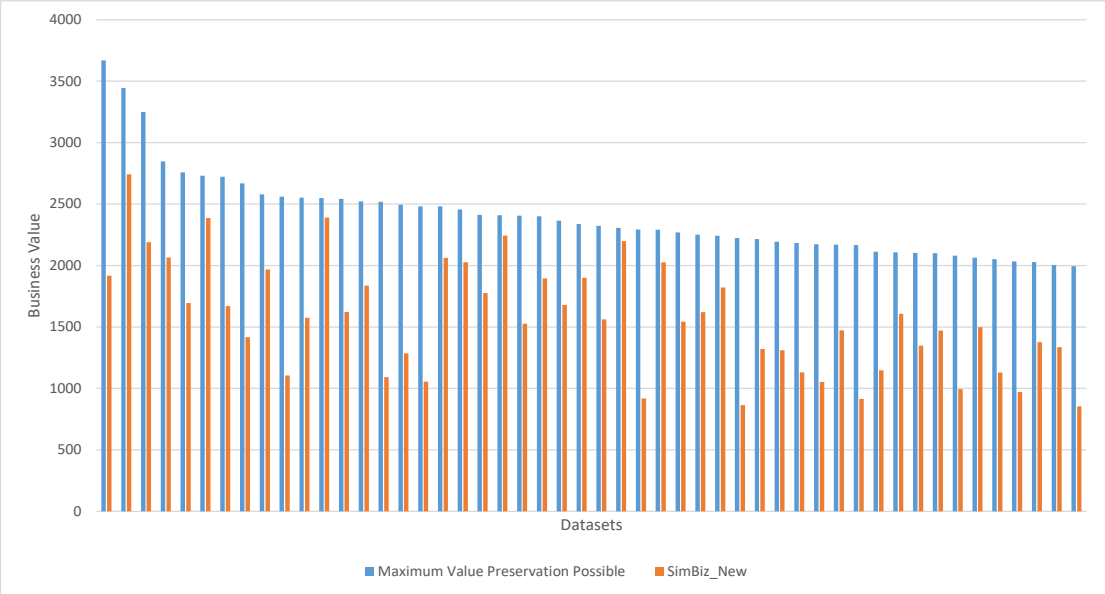


Figure 10. Maximum preservable value vs. SimBiz_New

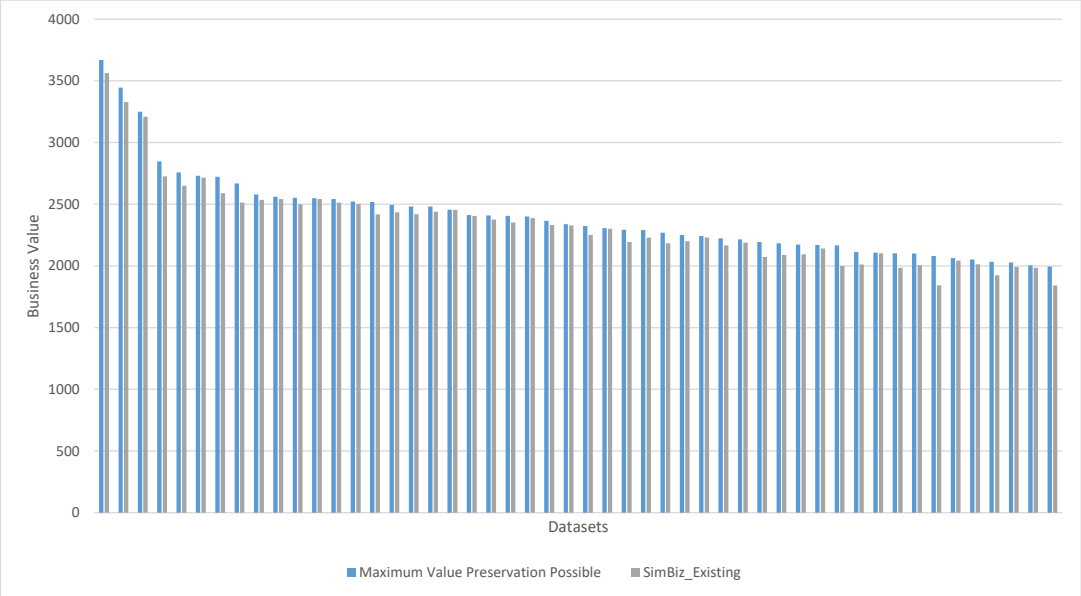


Figure 11. Maximum preservable value vs. SimBiz_Existing

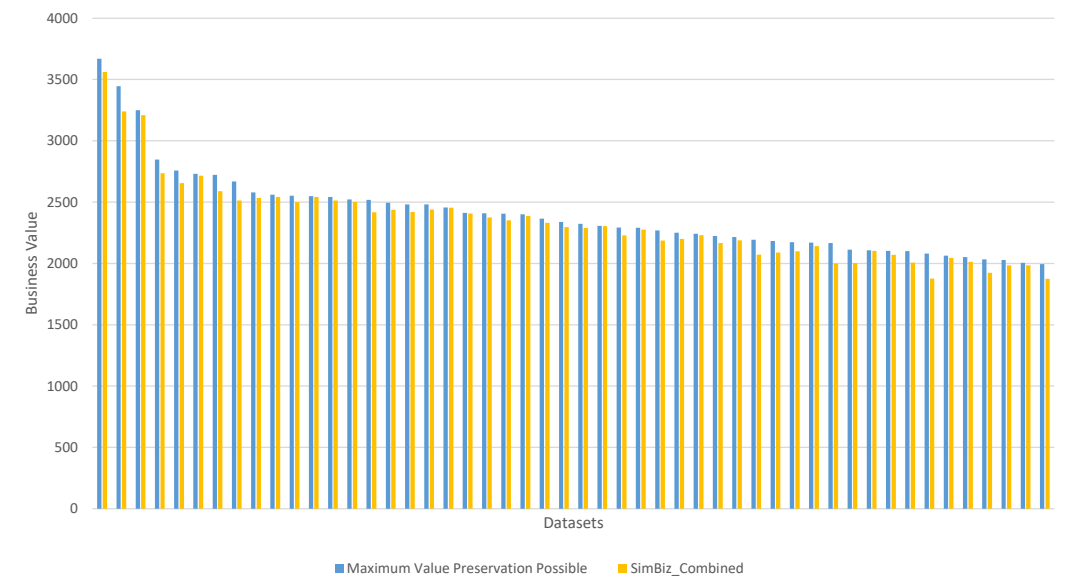


Figure 12. Maximum preservable value vs. SimBiz_Combined

From the boxplot in Figure 13 it can be observed that the configurations in which the business value of the user stories associated with the previous release(s) are considered for the computation of the importance-value perform better. From this observation, it can be concluded that considering the business value of the user stories pertaining to the previous release, in addition to the textual similarity, aids in preserving more value.

Figure 14 showcases a bar chart of the values preserved by the RSP, SimOnly, SimBiz_New, SimBiz_Existing, and SimBiz_Combined in comparison to the total business value and maximum preservable business value for 25 datasets out of the 200 datasets. The results for the rest of the datasets are given in Appendix B. Across all 200 datasets, SimBiz_New and SimOnly preserve almost equal amounts of business value. While examining the data, we observed that in a sample of 200 datasets, the average difference in value preservation was *36.64* compared to an average of the maximum preservable business value of 1264.39. Likewise, SimBiz_Existing and SimBiz_Combined consistently depict closely correlated values in terms of value preservation across the entirety of the dataset, with an average difference of *4.86*.

Based on the foregoing analysis of the results, it is evident that SimBiz_Existing and SimBiz_Combined exhibit superior performance relative to other configurations in terms of value preservation as seen in Figure 14. A plausible justification for this observed distinction is the incorporation of business values from user stories of the prior release into the prioritization procedure, as this is the only shared factor unique to these two configurations.

Figure 15 shows a comparison of performance by all the configuration in terms of percentage of value preserved (VP). The percentage of value preserved is computed with

respect to the maximum preservable value. From this figure it can again be observed that the configurations (SimBiz_Existing and SimBiz_Combined) considering business value of prior release user stories for prioritization preserved more value than the other configurations (RSP, SimOnly, and SimBiz_New). Figure 15 depicts the result for 25 datasets and the results for the remaining 175 datasets have been shown in Appendix B.

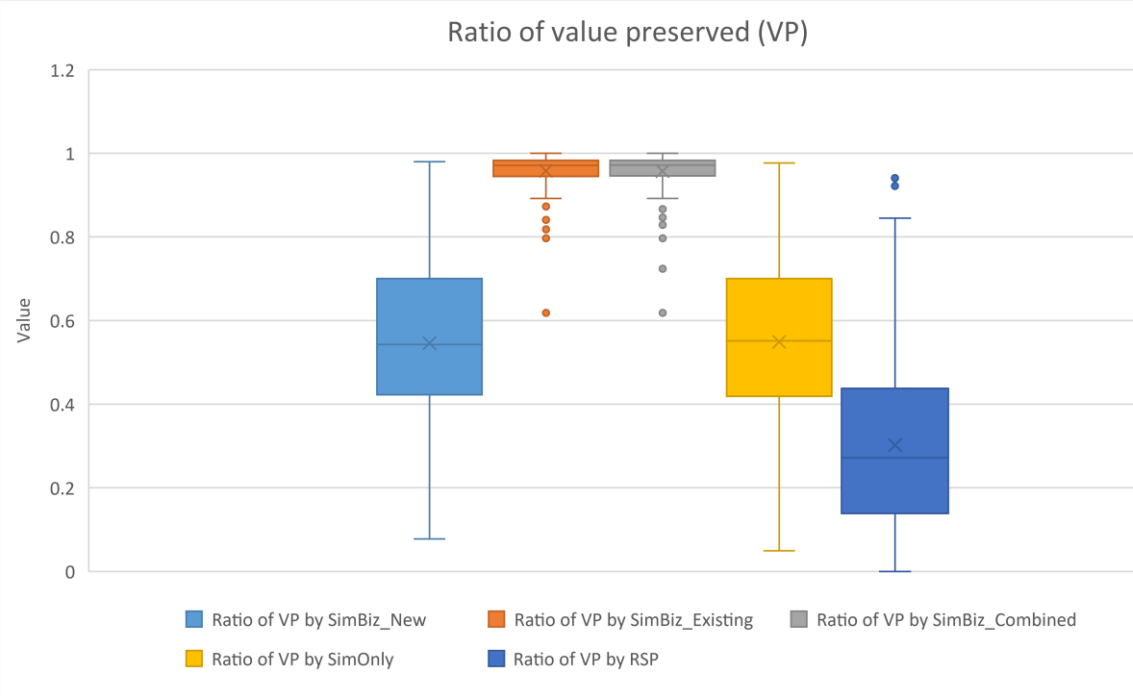


Figure 13. Box plot showing the performance of all the configurations.

ss

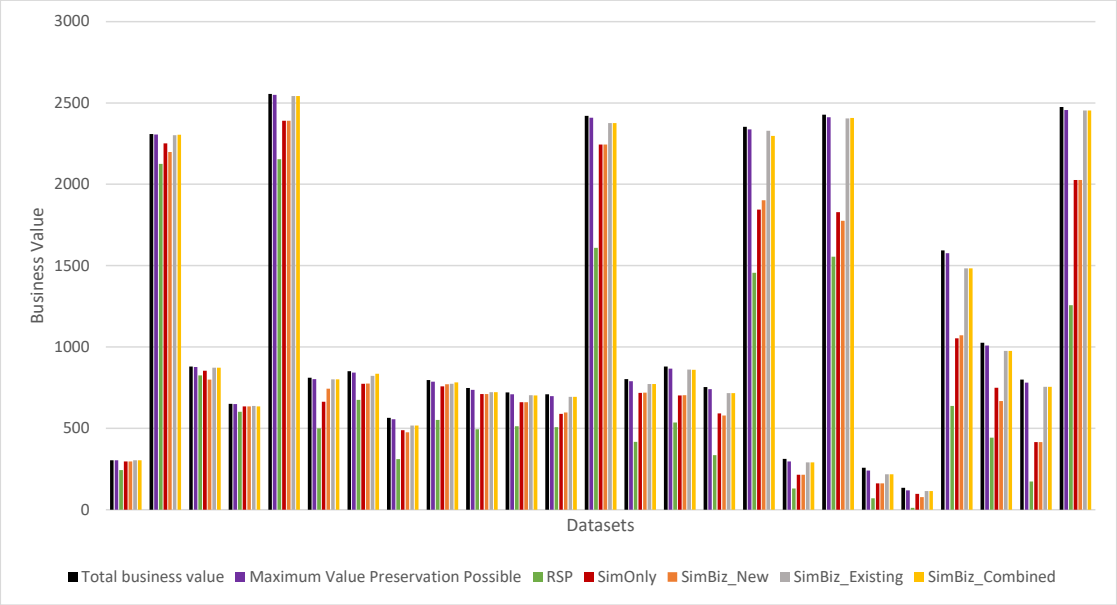


Figure 14. All configurations with respect to Total Business Value and Maximum Preservable Value

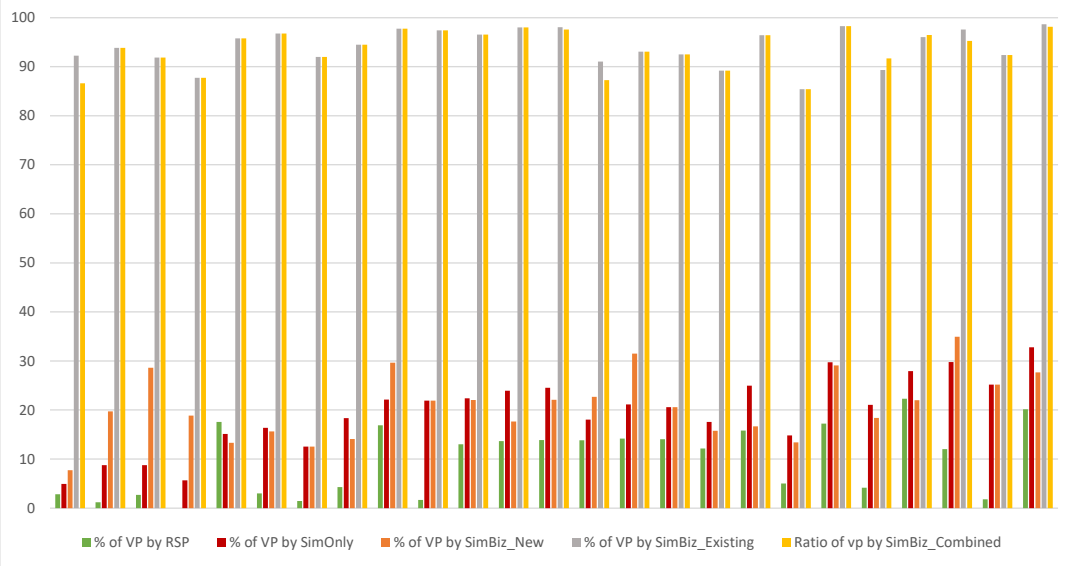


Figure 15. Percentage of Value Preserved by all configurations.

While the current results have shown valuable effectiveness it is crucial to acknowledge certain limitations inherent in the study.

### 5.2. Limitations:

In interpreting the results of this study, it is essential to acknowledge some inherent limitations. Firstly, the data synthesized for this research was prepared based on the researcher's perspective of the real world, possibly introducing subjective biases and potentially limiting the universality of our findings. Secondly, our text similarity analysis relied solely on the Spacy library. While efficient, other text similarity libraries might have rendered different results, leading to a more comprehensive and comparative analysis. Moreover, spaCy, a general-purpose text comparator may have inadvertently overlooked or misinterpreted specific domain-specific details, introducing potential errors. Such nuances emphasize the importance of training the language models with domain-specific data in future endeavors to ensure a more precise and accurate analysis within software engineering.

Based on the results observed in this chapter the following chapter dives deep to analyze these results. These analyses are then further used to answer the research questions presented in chapter 3.

# CHAPTER 6

# ANALYSIS AND DISCUSSION

In this chapter, we analyze and discuss the results presented in Chapter 5. This involves thoroughly examining the data to uncover patterns and correlations relevant to our research questions. We aim to provide insightful interpretations and contextualize the findings. This chapter explores the significance of the results and their implications for our research objectives.

## 6.1. Analysis:

Following are the two major analyses before getting into the discussion section where the research questions have been answered.

### 6.1.1. Analyzing the effect of business values from current release user stories:

From the discussions in the results section, Figure 14 and Figure 15 it can be observed that the inclusion of current release user stories for the computation of the importance-value did not create any impact in the value preservation. The configurations considering the business value of the current release user stories, i.e., SimBiz_New may have preserved more value than RSP for all 200 datasets but when compared to SimOnly, which did not have the factor of the business value it ended up preserving almost similar amount of value with a difference in the average value preservation of 1.13 as shown in Table 3. When the results of SimBiz_Combined were examined, it preserved a higher value than rest of the configurations (SimBiz_New, SimOnly, RSP) but showed very less variation in comparison to SimBiz_Existing. The only difference between SimBiz_Existing and SimBiz_Combined is again the inclusion of the business values of the current release user stories for the later. From both the above cases i.e., the comparison

between SimOnly and SimBiz_New and comparison between SimBiz_Existing and SimBiz_Combined it can be observed that the inclusion of business values from the current release user stories for prioritizing them before performing the selection process do not create any impact to preserve additional value through the test selection approach. To analyze further, the data was segregated into two sections based on the distribution of their business values. As mentioned in section 4.1.2, the first 100 datasets have uniform distribution, whereas the second 100 datasets have a right-skewed business value distribution. The average total business value for the second 100 datasets was much lower (703.71) compared to the first 100 datasets (2169.28). The results were analyzed for both cases to check for any variation compared to the complete set of 200 datasets. For the first 100 datasets, the percentage of average value preserved to the maximum preservable value by SimOnly and SimBiz_New are 59.18% and 59.3%, respectively, whereas by SimBiz_Existing and SimBiz_Combined are 96.39% and 96.4% respectively. For the second 100 datasets, where the total average business value was lower than the first 100 datasets, the percentage of average value preserved to the maximum preservable value by SimOnly and SimBiz_New are 56.34% and 55.6%, respectively, whereas by SimBiz_Existing and SimBiz_Combined are 96.24% and 96.22% respectively. This shows that even with the change in the business value distribution among the user stories, considering business value from current release user stories for prioritization does not impact value preservation.

### 6.1.2. Analyzing the effect of regression testing execution time:

Further we investigate the correlation between the percentage of selected test cases and the difference in the value preserved by the SimBiz_Existing and SimOnly. These two

configurations were considered as from analysis 1 it was seen that considering the business value of current release user story does not have much of an impact in value preservation. Thus, considering the configurations which do not consider business value of current release user stories was sufficient to understand the correlation between percentage of test cases selected and the value preserved by all the configurations. It was observed that with the increase in the percentage of selected test cases, the difference in value preserved between the various configurations decrease. To identify this trend, the correlation was computed between 2 sets of data. The correlation coefficient was found using the in-built Excel formula =CORREL(Series1, Series2) [48]. It calculates the correlation coefficient using the Pearson correlation coefficient. The correlation coefficient ranges from -1 to 1, where 1- indicates a perfect negative linear relationship, 0 indicates no linear relationship and 1 indicates a perfect positive linear relationship. The Pearson correlation coefficient is calculated using the below equation:

$$Correl(X,Y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2(y-\bar{y})^2}} \quad (10)$$

Where,

$Correl(X,Y)$ = correlation coefficient computed between series *X* and series *Y;*

$x$ = Values of the series *X;*

$\bar{x}$ = Mean of the values of series *X;*

$y$ = Values of the series *Y;*

$\bar{y}$ = Mean of the values of series *Y.*

The difference between the value preserved by SimBiz_Existing and SimOnly correlated to the percentage of selected test cases: The correlation coefficient for these two series of data across all 200 test cases came out to be -0.6579. A value of -0.6579 as

the correlation coefficient indicates a moderate negative linear relationship between the two data series, which means that the other tends to decrease as one variable increases.

The difference between the value preserved by SimBiz_Existing and Random selection correlated to the percentage of selected test cases: The calculated correlation coefficient for the two data series, considering all 200 datasets, is -0.37702. This correlation coefficient of -0.37702 denotes a perceptible negative linear relationship of weaker strength between the two data series, yet an increase in one variable is associated with a tendency for the other variable to decrease.

This observation is because, with the increase in the % of test cases selected, all configurations tend to preserve more values as more test cases associated with more user stories get tested. Thus, the difference in value preserved between the different configurations decrease, and all the configurations tend to be closer to the maximum preservable value. This throws light onto a scenario where almost all the test cases get selected, depicting a close resemblance to the case of *retest-all* i.e., running all the existing test cases for regression testing. Figure 16 shows the two ends of the percentage of test cases selected showcasing the values. Figure 17 is the bar chart showing the business value preservation by all the configurations in comparison to the total value preservation and maximum preservable value. This bar chart shows the effect on business value preservation by the change in percentage of test cases selected.

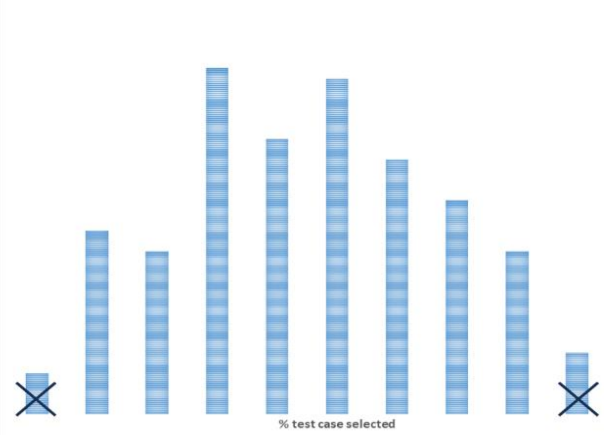


Figure 16. Plot showing higher and lower percentage of test case selected.

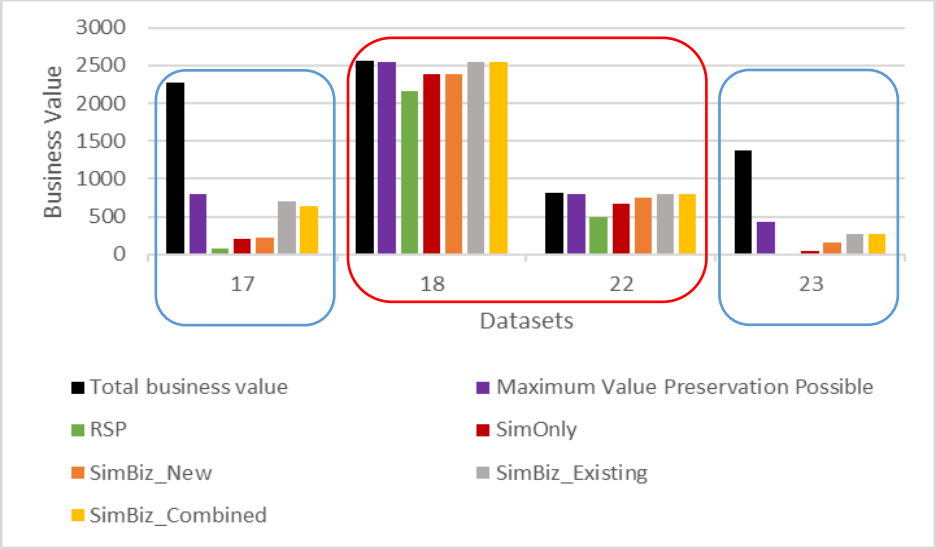


Figure 17: Value preservation based on percentage of test cases selected.

Based on the above analysis the following subsection addresses the research questions presented in chapter 3.

### 6.2. Discussion:

To address RQ1.1 we investigate the results in section 5.1.3 and Figure 10. SimBiz_New in our experiment considers similarity value in conjunction with the current release business value. For SimBiz_New the results show an average value preservation of 738.41 in comparison to 1264.39, the average maximum preservable business value across all 200 datasets, which is 58.40% of the maximum preservable value. When compared to SimOnly which refrains from using any business value an average value preservation of 739.54 (58.48% of the maximum preservable value) is noticed which is very similar to SimBiz_New. Though this configuration was seen to have performed better than Random selection process (average value preservation: 428.65 which is 33.90% of the maximum preservable value), the incorporation of the business of the current release user stories value in addition to text similarity value (when compared to SimBiz_New) does not show much of an impact. We believe, since the value preservation of the regression suite is computed on business values of the prior release user stories the inclusion of current release user stories into the approach of RTS does not create much of an impact.

This brings our discussion to answer the next research question, RQ1.2. SimBiz_Existing was built to analyze this research question. SimBiz_Existing uses the business values of prior release user stories along with the textual similarity value to prioritize the user stories before performing the test selection process. From the results in section 5.1.4 and Figure 11 we observe that SimBiz_Existing has preserved a substantial amount of average business value 1218.36 in comparison to the average maximum preservable business value of 1264.39 across all 200 datasets, which is 96.35% of the maximum preservable value. SimBiz_Existing was seen to have performed better than

Random selection process as well as the configurations which did not consider the business value of the prior release user stories. This can be reasoned due to the fact that inclusion of prior release user stories allows SimBiz_Existing to take an informed decision of prioritizing higher business value user stories from prior release, higher in the order. Thus, while selecting the test cases for regression suite the higher business user stories are selected first as RTS focuses on selecting test cases from prior release user stories solely. From this discussion it can be inferred that inclusion of business value from prior release user stories for prioritizing user stories can be a highly productive choice in terms of value preservation for an RTS approach.

Following the discussion of RQ1.2, RQ1.3 is attended which adds another factor, the business value of current release user stories to the factors of SimBiz_Existing, namely the textual similarity value and the business value of prior release user stories. SimBiz_Combined was implemented to address this research question. Results in 5.1.5 and Figure 12 show that the average value preservation by SimBiz_Combined is 1218.42 compared to the average of total available business value, 1264.39, which is 96.36% of the maximum preservable value. This is a substantially better value preservation when compared to the other configurations which did not include the business value of current release user stories. When compared to SimBiz_Existing (average value preservation: 1218.36 or 96.35%) which included the business value from prior release user story but did not include business value from current release user story, the value preservation data does not show much of a difference. From this result it can be inferred that the business value associated with the prior release user stories plays an important role in value preservation, whereas business value associated with current release user stories does not

have any impact on value preservation. This can also be observed in our discussion of RQ1.1 where SimBiz_New was compared to SimOnly.

To summarize RQ1, we derived that the inclusion of business value of prior release user stories can be very productive in preserving delivered value when implemented to prioritize user stories for an RTS approach but business value pertaining to current release user stories does not play much of a role in value preservation.

To answer RQ2.1 we consider the effect of equal execution time for each test case. The equal execution time for each test case does not create any impact on the different configurations used in this thesis. This is because there is no time-based weightage on the test cases while selecting the test cases. All the test cases are considered equal from the perspective of their execution time which does not add in as any additional factor while performing the selection process. The increase or decrease of this time can only affect the number of test cases getting selected for the regression suite when the total execution window for regression testing is constant. With the decrease in the execution time for all test cases, a higher number of test cases get selected for the regression suite which increases the value preserved by all the configurations and get closer to the total business value given the total execution window for the regression testing is constant. This is due to the increase in the percentage of selected test cases as discussed in Analysis 2 in this chapter.

For answering RQ2.2 the variation of the total available execution time for regression testing is considered given the execution time for each test case is constant. This variation impacts the total test cases that can be selected through the RTS approach. With the increase in the execution window all the configurations tend to preserve higher values as the percentage of selected test cases increases as shown in Figure 16 and 17. This is

because when a higher percentage of test cases are selected higher number of existing user stories get tested through regression testing and thus preserving higher value.

To summarize RQ2, time can create an impact on the value preserved by the RTS approach as it impacts the number of test cases getting selected for the regression suite.

# CHAPTER 7

# CONCLUSIONS AND FUTURE WORK

In this chapter, we draw conclusions from our analysis in Chapter 6 and outline avenues for future work. This chapter encapsulates the key insights derived from the study, summarizing the findings and their implications. Additionally, we discuss potential directions for future research, identifying areas that merit further exploration to advance the understanding and application of the study's outcomes.

## 7.1. Conclusion:

Regression testing is a critical part of software development to maintain the quality of the already implemented functionalities in the application while adding new improvements. Regression test selection has always been impactful in reducing time and effort for regression testing while ensuring the high quality of software. In this study, the technique selected a smaller set of test cases for regression and focused on testing existing user stories delivering high business value to the customers. Business value, as seen in the previous discussions, is a measure of the significance of each user story on the overall business objectives and goals.

User stories were investigated, and textual similarity was measured between the user stories to identify the impact of current-release user stories on prior-release user stories. Along with the measurement of textual similarity, the business values of user stories were considered to select the test cases using the RTS approach. The proposed RTS technique was validated by comparing to the Random Test Selection Process and with SimOnly which did not implement the factor of user story business value. Through the

experiments performed, it was observed and analyzed that the business value of current-release user stories does not impact value preservation as compared to the business value of prior-release user stories. Even with fewer test cases getting selected, a high level of value preservation was maintained when the business values of prior release user stories were considered in the configurations. This study also provides insights into the variation of the different factors involved in the computation of the RTS approach. The graphs and plots help us visualize the results and their effects compared to other configurations considering different factors in RTS process.

### 7.2. Future Work:

In exploring avenues for future work, there lies an exciting opportunity to enhance the efficiency and effectiveness of RTS. Varying execution time for individual test cases as mentioned in chapter 4 is an important factor to be considered when selecting test cases for regression suite. Execution time of test cases serves as a times-based weight that can be applied with other factors while performing RTS. Like the execution time of each test case, another test case level factor which can be crucial to make a well-informed decision for performing RTS is test case history. Regression suite is composed of test cases which have already been run once or more which provides us with a report of their history runs. This report provides information on how effective the test case has been in the past in identifying defects and how severe these defects were. This information can be used in addition to other factors to select more important test cases which can have the potential to find more critical defects.

This thesis focuses on identifying the impact of new changes based on the requirements and their textual similarities. One such factor which can be used to measure the impact of new changes on the existing functionalities is code modification. The use of code modification in addition to the similarity in requirements can be used to make more informed decisions in prioritizing user stories before selecting test cases for regression suite.

For textual similarity computation this thesis uses spaCy, a Natural Language Processing (NLP) tool. Though, en_core_web_lg, a specific model of spaCy used for the purpose of this work which have been seen to have performed well as discussed in section 4.2, training the model with domain-specific keywords could increase its efficiency.

In addition to these considerations, the prospect of incorporating machine learning techniques into the future development of RTS holds immense promise. Machine learning offers the potential for dynamic adaptability, predictive insights, and enhanced precision in the selection of regression test cases. Leveraging machine learning algorithms can empower the RTS process to autonomously adapt to evolving software landscapes, providing a more robust and efficient approach to test selection process.

# REFERENCES


[1] Rothermel, G. (1996). Efficient, effective regression testing using safe test selection techniques. Clemson University.

[2] Legunsen, O., Hariri, F., Shi, A., Lu, Y., Zhang, L., & Marinov, D. (2016, November). An extensive study of static regression test selection in modern software evolution. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 583-594).

[3] Kandil, P., Moussa, S., & Badr, N. (2017). Cluster-based test cases prioritization and selection technique for agile regression testing. Journal of Software: Evolution and Process, 29(6), e1794.

[4] Govil, N., & Sharma, A. (2021, October). A Game Plan to Build Optimized Regression Testing in Agile Methodologies Using Test Prioritization. In 2021 5th International Conference on Information Systems and Computer Networks (ISCON) (pp. 1-4). IEEE.

[5] Yu, T., & Wang, T. (2018, October). A study of regression test selection in continuous integration environments. In 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE) (pp. 135-143). IEEE.

[6] Rehan, M., Senan, N., Aamir, M., Samad, A., Husnain, M., Ibrahim, N., ... & Khatak, H. (2021). A systematic analysis of regression test case selection: a multi-criteria-based approach. Security and Communication Networks, 2021, 1-11.

[7] Suri, B., & Mangal, I. (2012). Analyzing test case selection using proposed hybrid technique based on BCO and genetic algorithm and a comparison with ACO. International Journal of Advanced Research in Computer Science and Software Engineering, 2(4).

[8] Yoo, S., & Harman, M. (2007, July). Pareto efficient multi-objective test case selection. In Proceedings of the 2007 international symposium on Software testing and analysis (pp. 140-150).

[9] Poppendieck, M., & Poppendieck, T. (2003). Lean software development: an agile toolkit. Addison-Wesley.

[10] Dini, N., Sullivan, A., Gligoric, M., & Rothermel, G. (2016, October). The effect of test suite type on regression test selection. In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (pp. 47-58). IEEE.

[11] Rothermel, G., & Harrold, M. J. (1996). Analyzing regression test selection techniques. IEEE Transactions on software engineering, 22(8), 529-551.

[12] Zhang, L. (2018, May). Hybrid regression test selection. In Proceedings of the 40th International Conference on Software Engineering (pp. 199-209).

[13] Azizi, M., & Do, H. (2018, October). Retest: A cost effective test case selection technique for modern software development. In 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE) (pp.144-154). IEEE.

[14] Kaur, A., & Agrawal, A. P. (2017, January). A comparative study of bat and cuckoo search algorithm for regression test case selection. In 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence (pp. 164-170). IEEE.

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

[16] Srivastva, P. R., Kumar, K., & Raghurama, G. (2008). Test case prioritization based on requirements and risk factors. *ACM SIGSOFT Software Engineering Notes*, *33*(4), 1-5.

[17] Wang, X., & Zeng, H. (2016, May). History-based dynamic test case prioritization for requirement properties in regression testing. In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery* (pp. 41-47).

[18] Chittimalli, P. K., & Harrold, M. J. (2008, February). Regression test selection on system requirements. In *Proceedings of the 1st India software engineering conference* (pp. 87-96).

[19] Arafeen, M. J., & Do, H. (2013, March). Test case prioritization using requirements-based clustering. In 2013 IEEE sixth international conference on software testing, verification and validation (pp. 312-321). IEEE.

[20] Suri, B., & Singhal, S. (2011). Implementing ant colony optimization for test case selection and prioritization. International journal on computer science and engineering, 3(5), 1924-1932.

[21] Kazmi, R., Jawawi, D. N., Mohamad, R., & Ghani, I. (2017). Effective regression test case selection: A systematic literature review. ACM Computing Surveys (CSUR), 50(2), 1-32.

[22] Malishevsky, A. G., Rothermel, G., & Elbaum, S. (2002, October). Modeling the cost-benefits tradeoffs for regression testing techniques. In *International Conference on Software Maintenance, 2002. Proceedings.* (pp. 204-213). IEEE.

[23] Khalid, Z., & Qamar, U. (2019, October). Weight and cluster based test case prioritization technique. In 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 1013-1022). IEEE.

[24] Bach, J. (1999). Risk and requirements-based testing. Computer, 32(06), 113-114.

[25] Panigrahi, C. R., & Mall, R. (2014). A heuristic-based regression test case prioritization approach for object-oriented programs. *Innovations in Systems and Software Engineering*, *10*, 155-163.

[26] Khatibsyarbini, M., Isa, M. A., Jawawi, D. N., & Tumeng, R. (2018). Test case prioritization approaches in regression testing: A systematic literature review. Information and Software Technology, 93, 74-93.

[27] J. Anderson, H. Do, and S. Salem. Customized regression testing using telemetry usage patterns. In Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on. IEEE, 2016.

[28] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel. The effects of time constraints on test case prioritization: A series of controlled experiments. 36(5), 2010.

[29] T. Bin Noor and H. Hemmati. A similarity-based approach for test case prioritization using historical failure data. In International Conference in Software Reliability Engineering (ISSRE). IEEE, 2015.

[30] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability, 22(2):67–120, 2012.

[31] Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. Software testing, verification and reliability, 22(2), 67-120.

[32] K. K. Aggrawal, Yogesh Singh, and A. Kaur. Code coverage based technique for prioritizing test cases for regression testing. In ACM SIGSOFT Software Engineering Notes. ACM, 2004.

[33] R. Carlson, H. Do, , and A. Denton. A clustering approach to improving test case prioritization: An industrial case study. In ICSM ’11 Proceedings of the 2011 27th IEEE International Conference on Software Maintenanc, pages 382–391. IEEE-ACM, 2011.

[34] S. Elbaum, A. G. Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering, 28(2):159–182, February 2002.

[35] H. Mei, D. Hao, L. Zhang, L. Zhang, J. Zhou, and G. Rothermel. A static approach to prioritizing junit test cases. IEEE Transactions on Software Engineering, 38(6):1258–1275, 2012.

[36] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability, 22(2):67–120, 2012.

[37] L. Zhang, D. Hao, L. Zhang, G. Rothermel, and H. Mei. Bridging the gap between the total and additional test-case prioritization strategies. In Proceedings of ICSE, 2013.

[38] Spieker, H., Gotlieb, A., Marijan, D., & Mossige, M. (2017, July). Reinforcement learning for automatic test case prioritization and selection in continuous integration. In Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 12-22).

[39] Achimugu, P., Selamat, A., Ibrahim, R., & Mahrin, M. N. R. (2014). A systematic literature review of software requirements prioritization research. *Information and software technology*, *56*(6), 568-585.

[40] Hudaib, A., Masadeh, R., Qasem, M. H., & Alzaqebah, A. (2018). Requirements prioritization techniques comparison. *Modern Applied Science*, *12*(2), 62.

[41] Ma, Q. (2009). *The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review* (Doctoral dissertation, Auckland University of Technology).

[42] Sureshchandra, K., & Shrinivasavadhani, J. (2008, August). Moving from waterfall to agile. In *Agile 2008 conference* (pp. 97-101). IEEE.

[43] Malhotra, C., & Chug, A. (2013). Agile testing with Scrum-A survey. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(3), 452-459.

[44] Harish, R., Madhu, B. K., & Lokesha, V. (2012). A sophisticated study on best practices of agile software testing. *International Journal of Electronics communication and computer engineering*, *3*(1), 30.

[45] Hellmann, T. D., Sharma, A., Ferreira, J., & Maurer, F. (2012, August). Agile Testing: Past, Present, and Future--Charting a Systematic Map of Testing in Agile Software Development. In *2012 Agile Conference* (pp. 55-63). IEEE.

[46] Ali, S., Hafeez, Y., Hussain, S., & Yang, S. (2020). Enhanced regression testing technique for agile software development and continuous integration strategies. Software Quality Journal, 28, 397-423.

[47] Koivuniemi, J. (2017). Shortening feedback time in continuous integration environment in large-scale embedded software development with test selection (Master's thesis, J. Koivuniemi).

[48] Haghighatkhah, A., Mäntylä, M., Oivo, M., & Kuvaja, P. (2018). Test prioritization in continuous integration environments. Journal of Systems and Software, 146, 80-98.

[49] Elbaum, S., Rothermel, G., & Penix, J. (2014, November). Techniques for improving regression testing in continuous integration development environments. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 235-245).

[50] Shi, A., Zhao, P., & Marinov, D. (2019, October). Understanding and improving regression test selection in continuous integration. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 228-238). IEEE.

[51] Yu, T., & Wang, T. (2018, October). A study of regression test selection in continuous integration environments. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 135-143). IEEE.

[52] Pan, R., Bagherzadeh, M., Ghaleb, T. A., & Briand, L. (2022). Test case selection and prioritization using machine learning: a systematic literature review. Empirical Software Engineering, 27(2), 29.

[53] SOLAR-group. (n.d.). GitHub - SOLAR-group/TAWOS: The Tawosi Agile Web-hosted Open-Source Issues dataset. GitHub. https://github.com/SOLAR-group/TAWOS

[54] Palma, F., Abdou, T., Bener, A., Maidens, J., & Liu, S. (2018, October). An improvement to test case failure prediction in the context of test case prioritization. In Proceedings of the 14th international conference on predictive models and data analytics in software engineering (pp. 80-89).

[55] Mahdieh, M., Mirian-Hosseinabadi, S. H., Etemadi, K., Nosrati, A., & Jalali, S. (2020). Incorporating fault-proneness estimations into coverage-based test case prioritization methods. Information and Software Technology, 121, 106269.

[56] Jugran, S., Kumar, A., Tyagi, B. S., & Anand, V. (2021, March). Extractive automatic text summarization using SpaCy in Python & NLP. In 2021

International conference on advance computing and innovative technologies in engineering (ICACITE) (pp. 582-585). IEEE.

[57] Pauzi, Z., & Capiluppi, A. (2020). Text similarity between concepts extracted from source code and documentation. In Intelligent Data Engineering and Automated Learning–IDEAL 2020: 21st International Conference, Guimaraes, Portugal, November 4–6, 2020, Proceedings, Part I 21 (pp. 124-135). Springer International Publishing.

[58] Jongeling, R., Datta, S., & Serebrenik, A. (2015, September). Choosing your weapons: On sentiment analysis tools for software engineering research. In 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 531-535). IEEE.

[59] Novielli, N., Calefato, F., & Lanubile, F. (2015, September). The challenges of sentiment detection in the social programmer ecosystem. In Proceedings of the 7th international workshop on social software engineering (pp. 33-40).

[60] Pletea, D., Vasilescu, B., & Serebrenik, A. (2014, May). Security and emotion: sentiment analysis of security discussions on github. In Proceedings of the 11th working conference on mining software repositories (pp. 348-351).

[61] Al Omran, F. N. A., & Treude, C. (2017, May). Choosing an NLP library for analyzing software documentation: a systematic literature review and a series of experiments. In 2017 IEEE/ACM 14th international conference on mining software repositories (MSR) (pp. 187-197). IEEE.

[62] Haris, M. S., Kurniawan, T. A., & Ramdani, F. (2020). Automated features extraction from software requirements specification (SRS) documents as the basis of software product line (SPL) engineering. Journal of Information Technology and Computer Science, 5(3), 279-292.

[63] Robeer, M., Lucassen, G., Van Der Werf, J. M. E., Dalpiaz, F., & Brinkkemper, S. (2016, September). Automated extraction of conceptual models from user stories via NLP. In 2016 IEEE 24th international requirements engineering conference (RE) (pp. 196-205). IEEE.

[64] Lucassen, G., Robeer, M., Dalpiaz, F., Werf, J.M., & Brinkkemper, S. (2017). Extracting conceptual models from user stories with Visual Narrator. Requirements Engineering, 22, 339-358.

[65] Kim, J. M., & Porter, A. (2002, May). A history-based test prioritization technique for regression testing in resource constrained environments. In Proceedings of the 24th international conference on software engineering (pp. 119-129).

[66] Chauhan, C., Dakoliya, M., & Sharma, R. K. (2023). Use of Fibonacci Sequence in Project Estimation. Indian Journal of Science and Technology, 16(33), 2649-2652.

[67] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.openai.com/chat

# APPENDIX A

# RESEARCH ARTIFACTS

**Github Link:** AMondal. (n.d.). GitHub - amondal8/masters-thesis. GitHub. https://github.com/amondal8/masters-thesis

This thesis focuses on prioritizing existing user stories based on the textual similarity with new user stories and the business values before selecting the associated test cases for regression test selection (RTS) process. Follow the folder **"Definitional data"** to access all the python files.

The results (.xlsx) of this thesis and the .sql containing the created database can be found in : **Thesis/Files/Definitional data/Additional Files**

**Understanding the Database:** The user stories have been extracted from the TAWOS dataset (https://github.com/SOLAR-group/TAWOS) and the remaining data have been synthesized. We have used two schemas while creating the data and storing the results. Below are the specifications of the schemas:

1. **Definitionaldata:** This schema is used to store the initial data before creating any specific datasets. The below table shows the data tables, their purpose and the fields used.
2. **Dataset_schema:** This schema is used to store datasets created from the data in definitionaldata schema. These are used to run the simulations. The below table shows the data tables, their purpose and the fields used.

**Understanding the Codebase:**

1. **Creating the tables:** All the tables for both the schemas have been created using our python codes. This action can also be done without the use of any programming code using MySQL Workbench (https://www.mysql.com/products/workbench/) or any similar tool.

2. Filling the tables with data:
   a. **Filling the tables of definitionaldata schema:** The extracted user stories from the TAWOS dataset are imported into our "userstory" table. The data are manually extracted from the TAWOS database into an excel and then the code is run to import the data to our table. The test case ids are also imported into the "testcase" table from an excel. Methods filling data into additional tables can be ignored for the purpose of this thesis and have been placed for future use.

   b. **Filling the tables of dataset_schema:** This involves three major parts:
      (i) **Creating dataset id:** Creating a record in the dataset table with a unique id to fill data into table specific to each dataset id.
      (ii) **Filling up the userstory_datasettable and tc_datasettable:** Data from the userstory and testcase tables are fetched and filled into the userstory_datasettable and tc_datasettable respectively specific to every dataset id based on the user provided input (count of user stories in each

release and count of test cases, for a given dataset). The user inputs can be provided using the configuration file or the data.py file based on the requirement.

(iii) **Mapping user stories to test cases:** An adjacency matrix is created based on the total number of user stories and test cases available for a given dataset id to create a mapping. This mapping is then stored to us_tc_map table specific to every dataset id.

3. **Running the Simulations:** Once the data has been created and pushed to the tables in accordance to specific datasets (ds_id) the final simulation is run. Running the final file produces the results specific to every dataset and the results are stored into the dataset table of dataset_schema. Along with storing the results to the database they are also exported to the desired excel into specific columns. The results in the excel are used to visualize the results and dive deeper into analyzing them.

**Steps to be followed to run the simulations:**

The above processes have been mentioned in a chronological order which has to be followed to run the simulations and replicate the results of this thesis. Following are the steps revisited for better understanding.

*Step1:* Extract the textual content (user story description) from the TAWOS dataset and place it on an .xlsx file which will be used to import these to our database.

**Note:** Prerequisite for Step2 and Step3: The schemas need to be created manually in the database and updated in the "createTables.py" and "createInstanceTables.py" files.

*Step2:* Run the createTables.py to create the tables in "definitionaldata" schema (if not already created).

*Step3:* Run the createInstanceTables.py to create the tables in "dataset_schema" schema (if not already created).

*Step4:* Run the fill_primarytable.py to fill in data to the tables "userstory" and "testcases" of the "definitionaldata" schema. These data are extracted from the .xlsx file mentioned in Step1 and imported to our database.

*Step5:* Run the config_initialsetup.py to create a new dataset id corresponding to which the following data will be created. This creates a new id everytime it is run.

*Step6:* Run the fill_datasettable.py to categorize the user stories into releases and fills in the "userstory_datasettable" table. The count of user stories in each release is user defined and can be changed for each release. This also imports the user defined count of test cases into the "tc_datasettable" table. All the data are imported from "userstory" and "testcases" tables of the "definitionaldata" schema.

*Step7:* Run the fill_mappingtable to map the user stories to test cases and store them in the "us_tc_map" table.

*Step8:* Once all the tables are filled run the final_implementation1.py to fetch the results onto the dataset table and the desired .xlsx file.

Following the steps should provide the results we obtained for this thesis.

# APPENDIX B

# EXPERIMENTAL RESULTS

Figures 18 to 21 show results of 200 datasets for Random Selection Process (RSP):

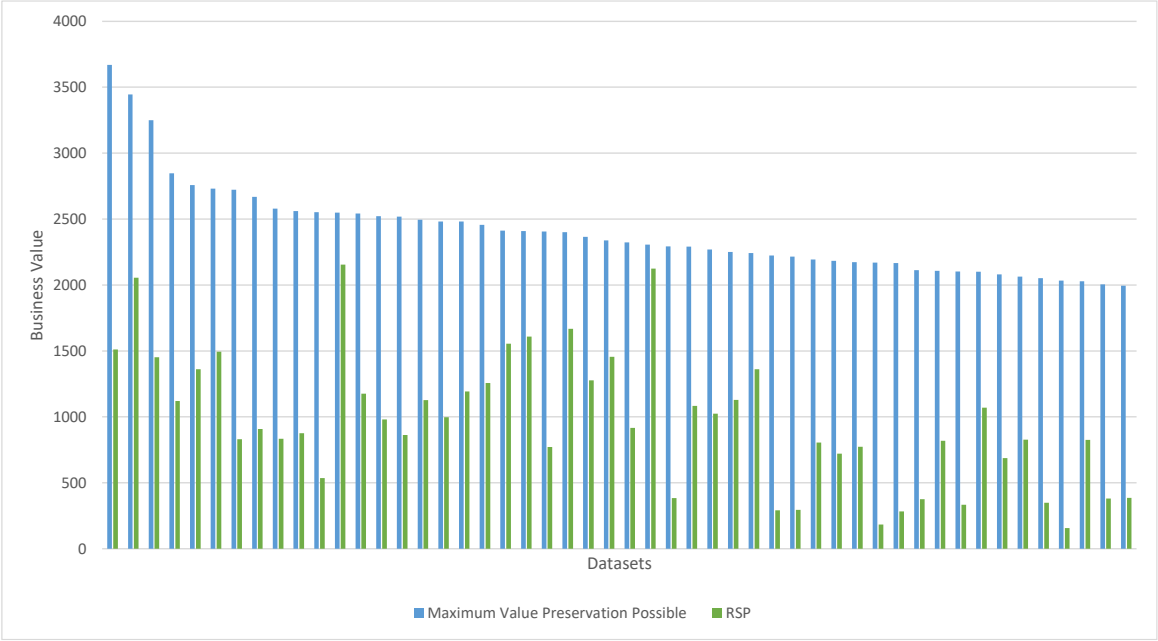


Figure 18: Maximum Preservable Business Value vs. RSP-Plot1

(dataset range: 1 - 50).


Figure 19: Maximum Preservable Business Value vs. RSP-Plot2

(dataset range: 51 - 100).

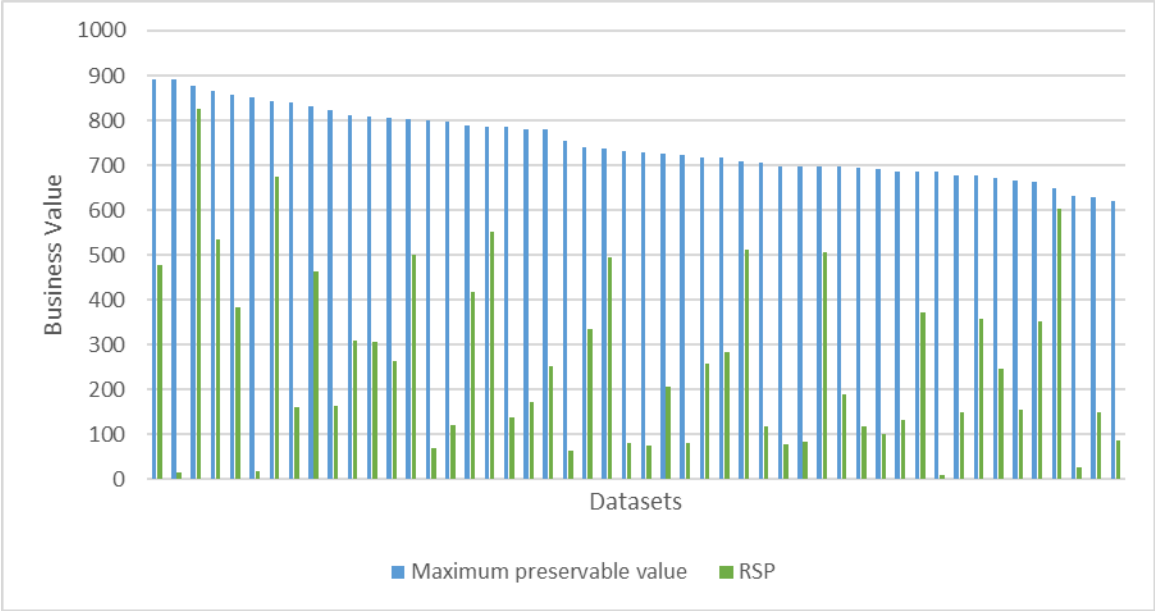


Figure 20: Maximum Preservable Business Value vs. RSP-Plot3

(dataset range: 101 - 150).

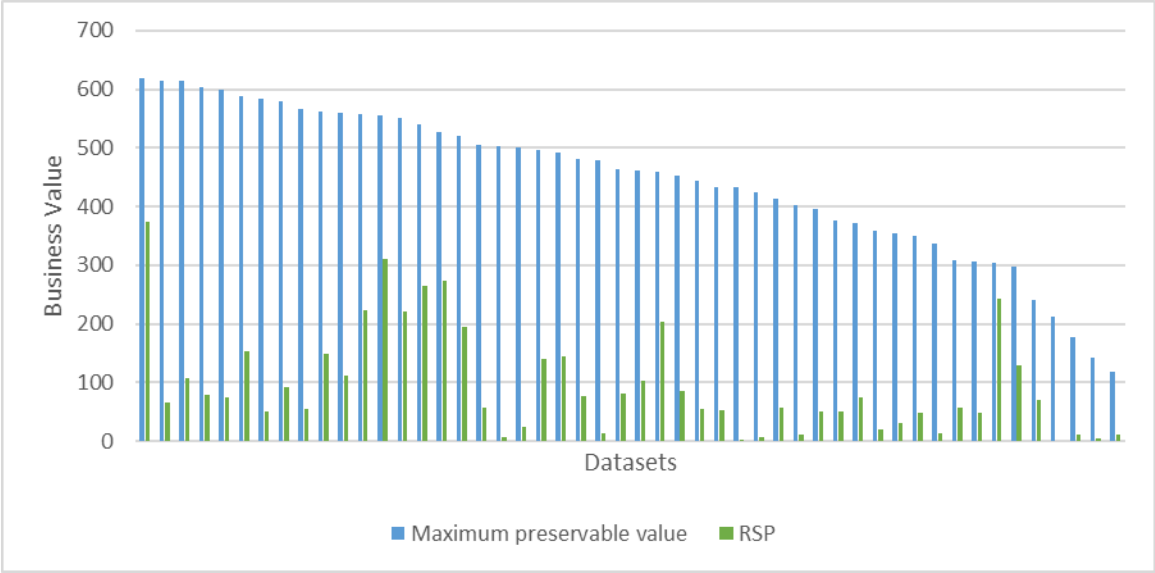


Figure 21: Maximum Preservable Business Value vs. RSP-Plot4

(dataset range: 151 - 200).

Figure 22 to 25 show results of 200 datasets for SimOnly:

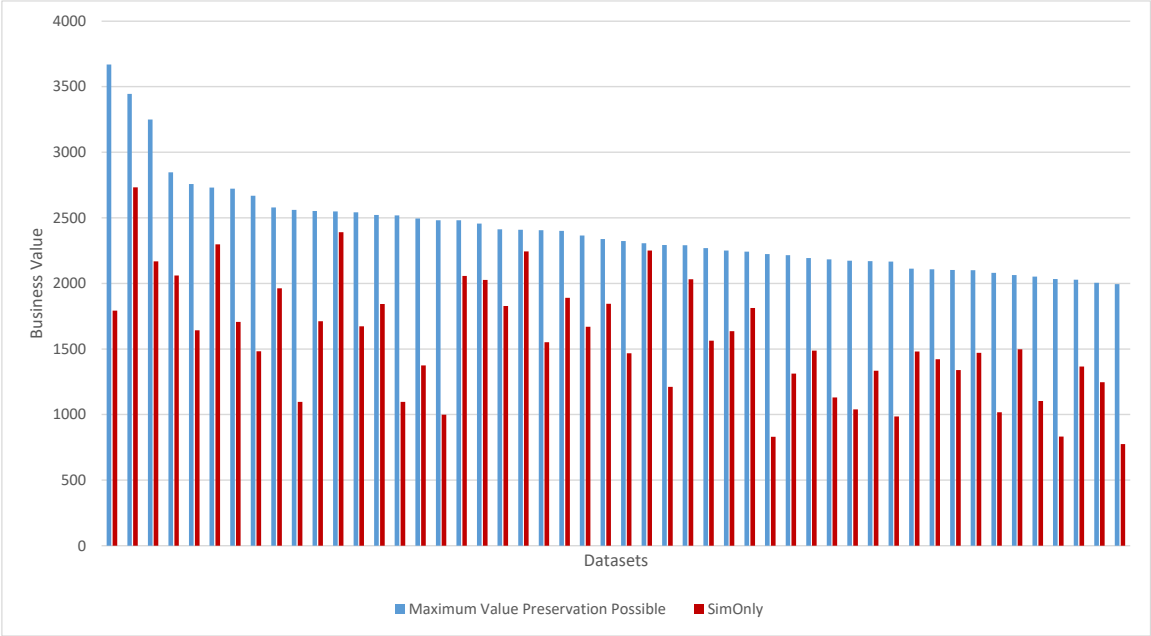


Figure 22: Maximum Preservable Business Value vs. SimOnly-Plot1

(dataset range: 1 - 50).

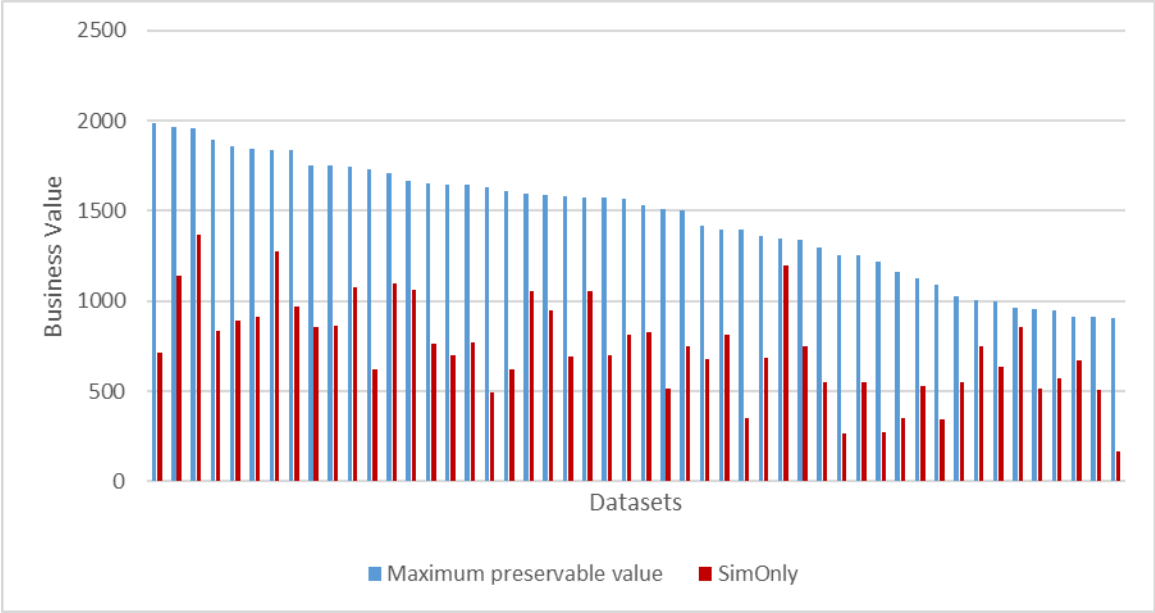


Figure 23: Maximum Preservable Business Value vs. SimOnly-Plot2

(dataset range: 51 - 100).

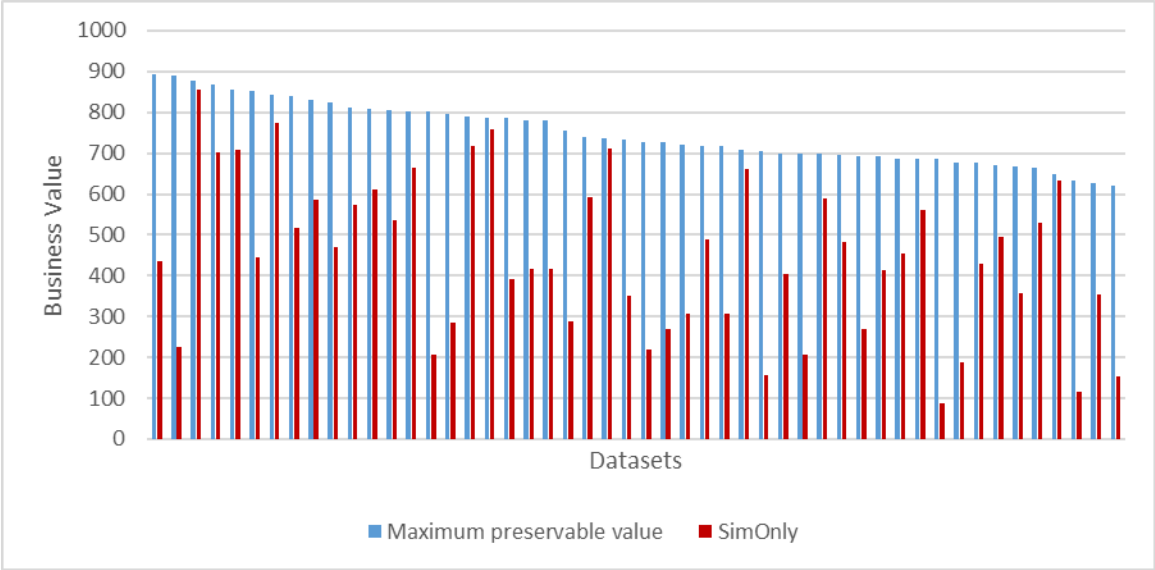


Figure 24: Maximum Preservable Business Value vs. SimOnly-Plot3

(dataset range: 101 - 150).

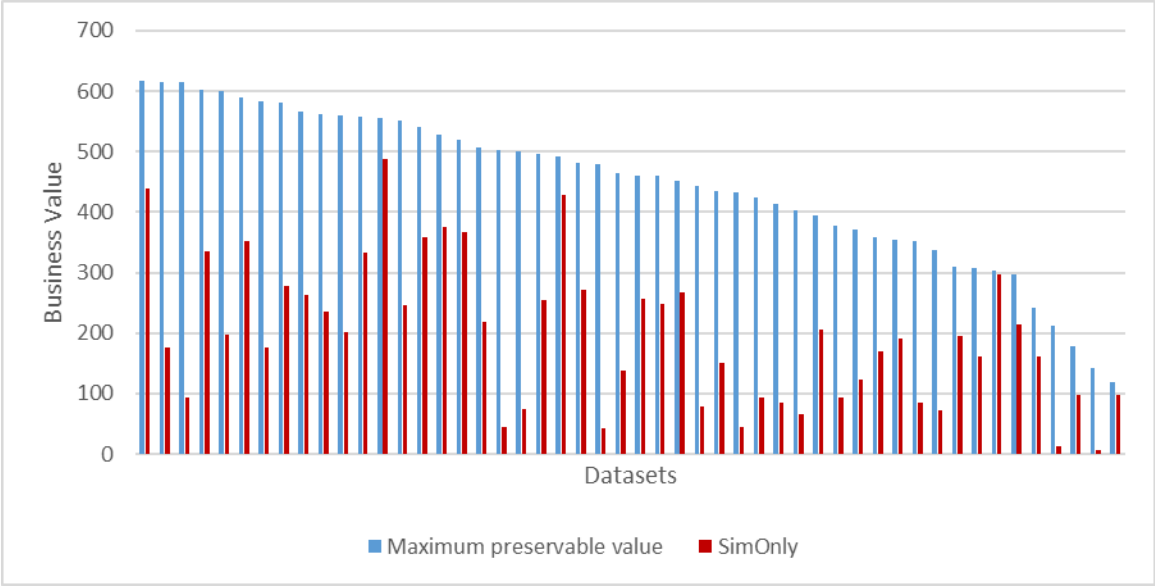


Figure 25: Maximum Preservable Business Value vs. SimOnly-Plot4

(dataset range: 151 - 200).

Figure 26 to 29 show results of 200 datasets for SimBiz_New:

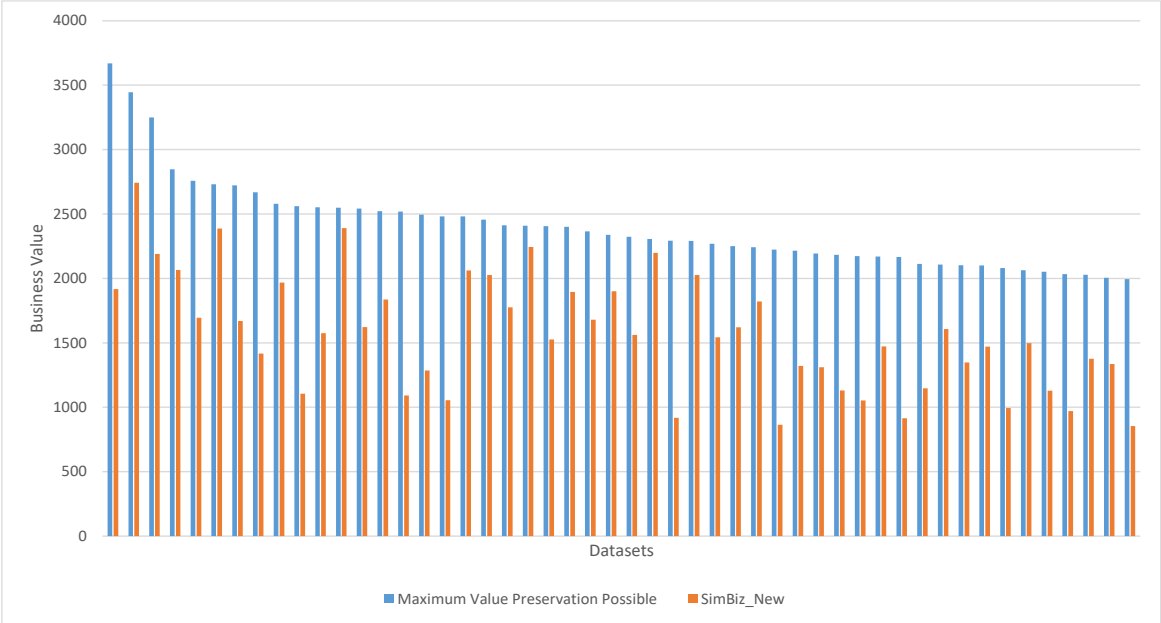


Figure 26: Maximum Preservable Business Value vs. SimBiz_New-Plot1 (dataset range: 1 - 50).

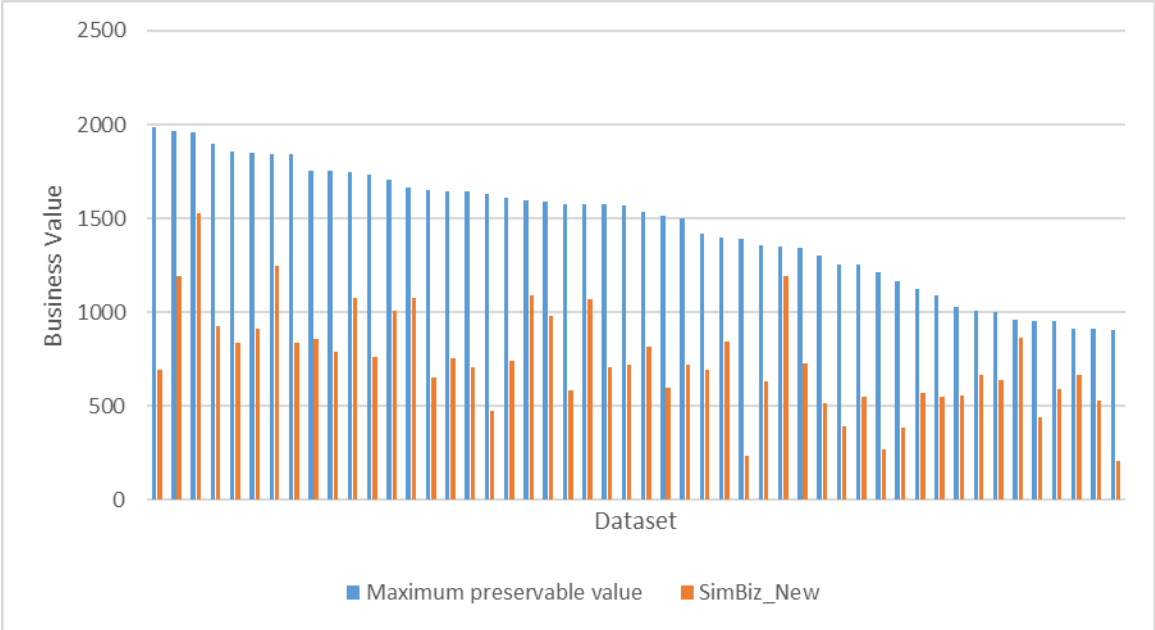


Figure 27: Maximum Preservable Business Value vs. SimBiz_New-Plot2 (dataset range: 51 - 100).

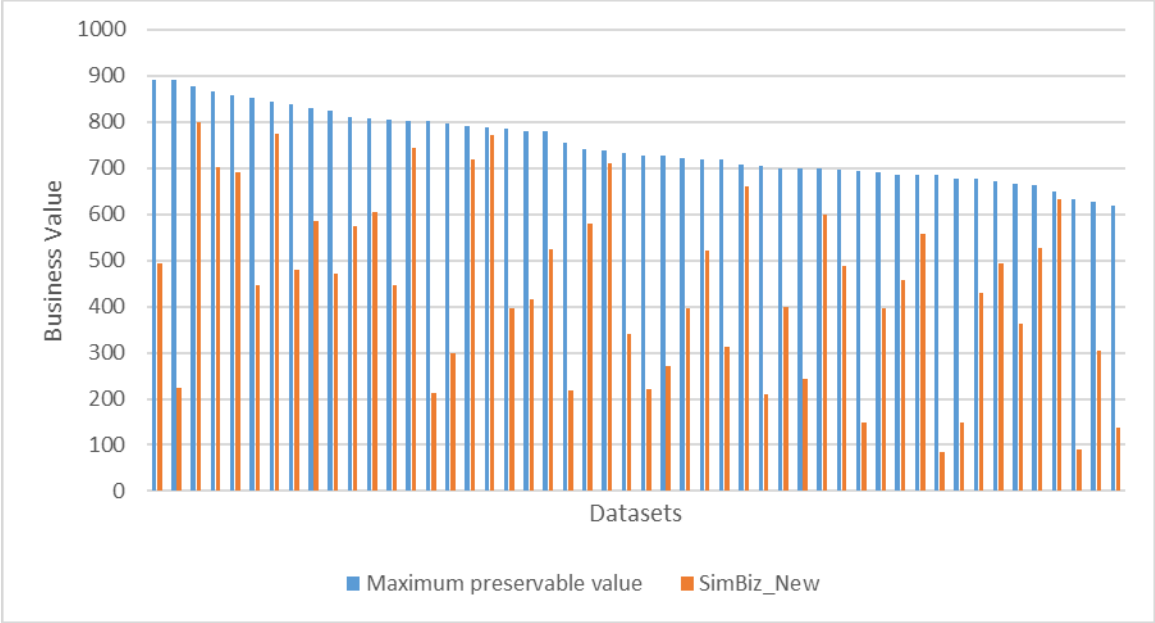


Figure 28: Maximum Preservable Business Value vs. SimBiz_New-Plot3 (dataset range: 101 - 150).

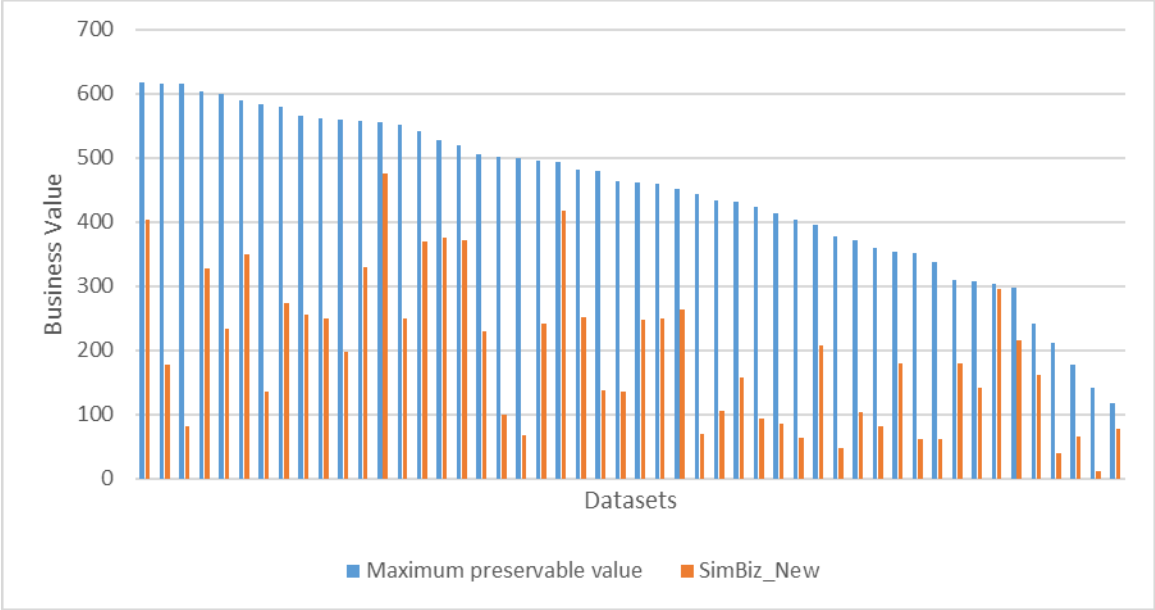


Figure 29: Maximum Preservable Business Value vs. SimBiz_New-Plot4 (dataset range: 151 - 200).

Figure 30 to 33 show results of 200 datasets for SimBiz_Existing:

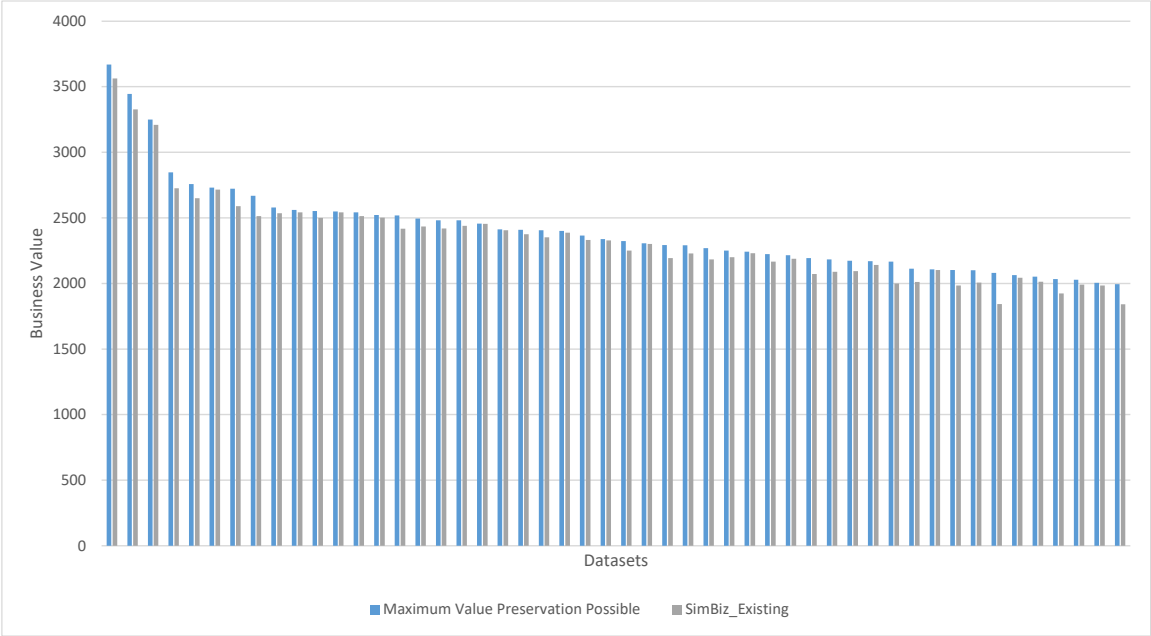


Figure 30: Maximum Preservable Business Value vs. SimBiz_Existing-Plot1 (dataset range: 1 - 50).

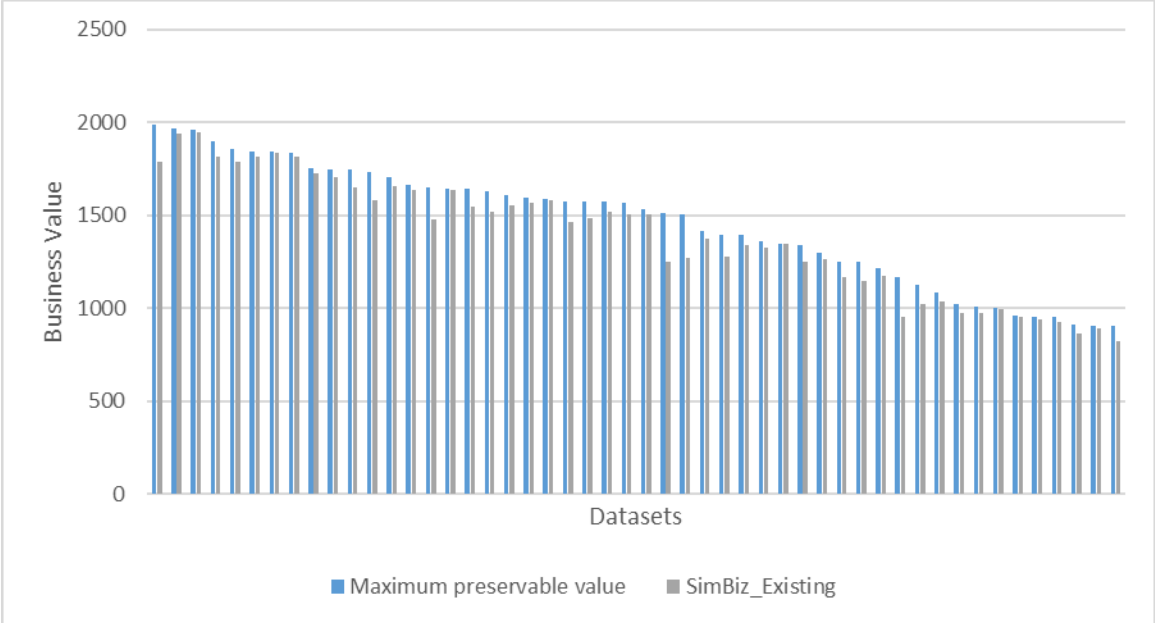


Figure 31: Maximum Preservable Business Value vs. SimBiz_Existing-Plot2 (dataset range: 51 - 100).

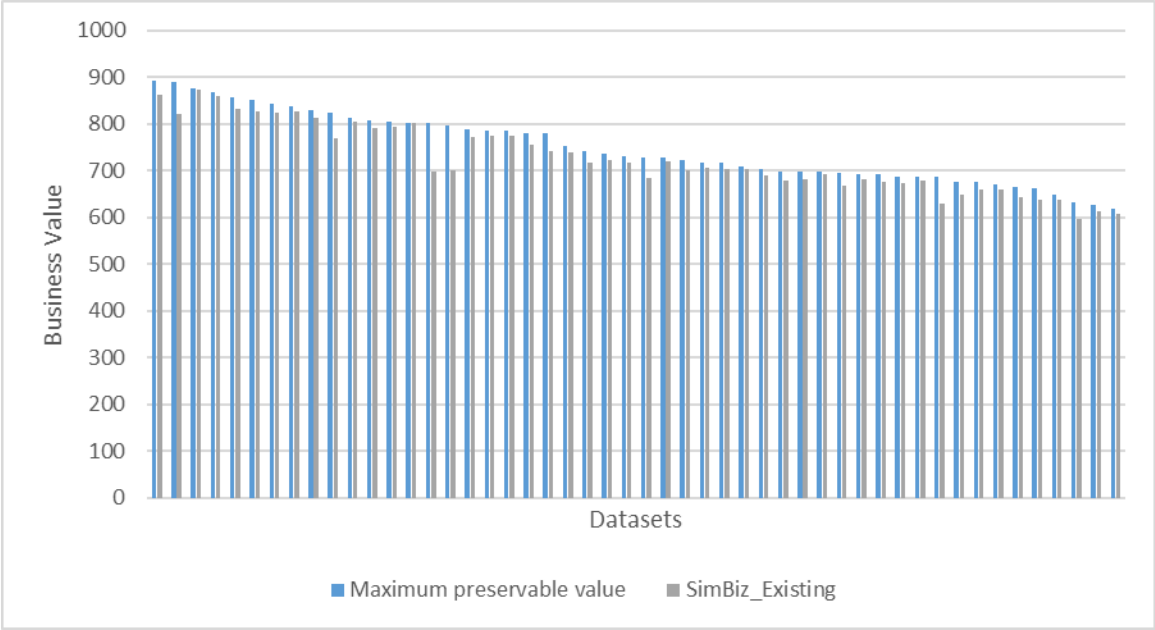


Figure 32: Maximum Preservable Business Value vs. SimBiz_Existing-Plot3 (dataset range: 101 - 150).

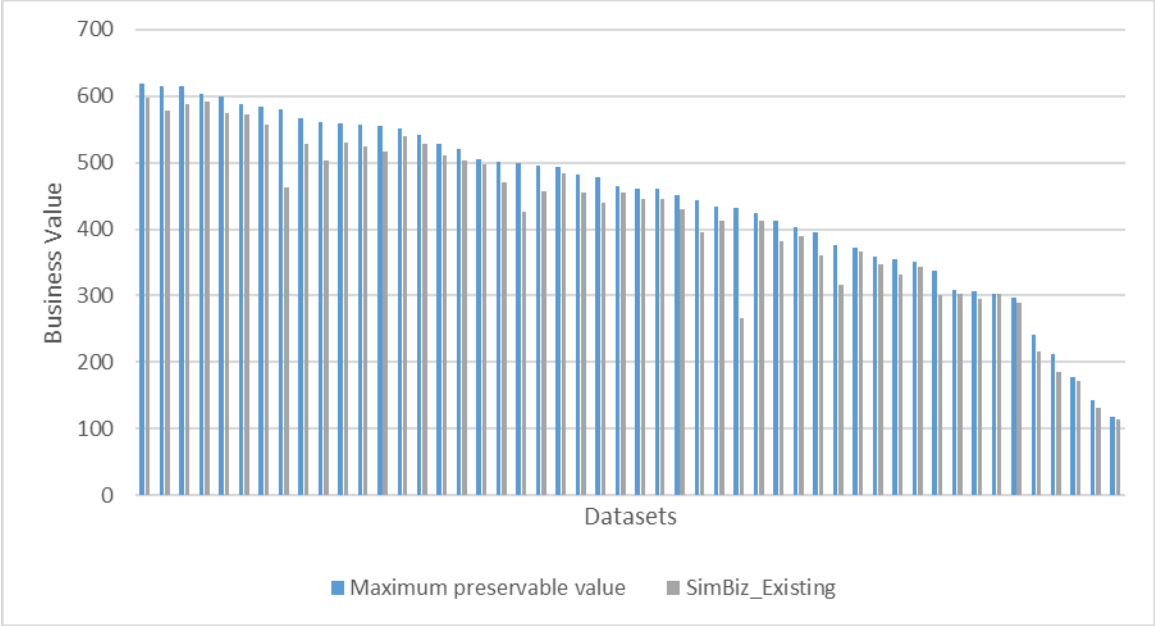


Figure 33: Maximum Preservable Business Value vs. SimBiz_Existing-Plot4 (dataset range: 151 - 200).

Figure 34 to 37 show results of 200 datasets for SimBiz_Combined:

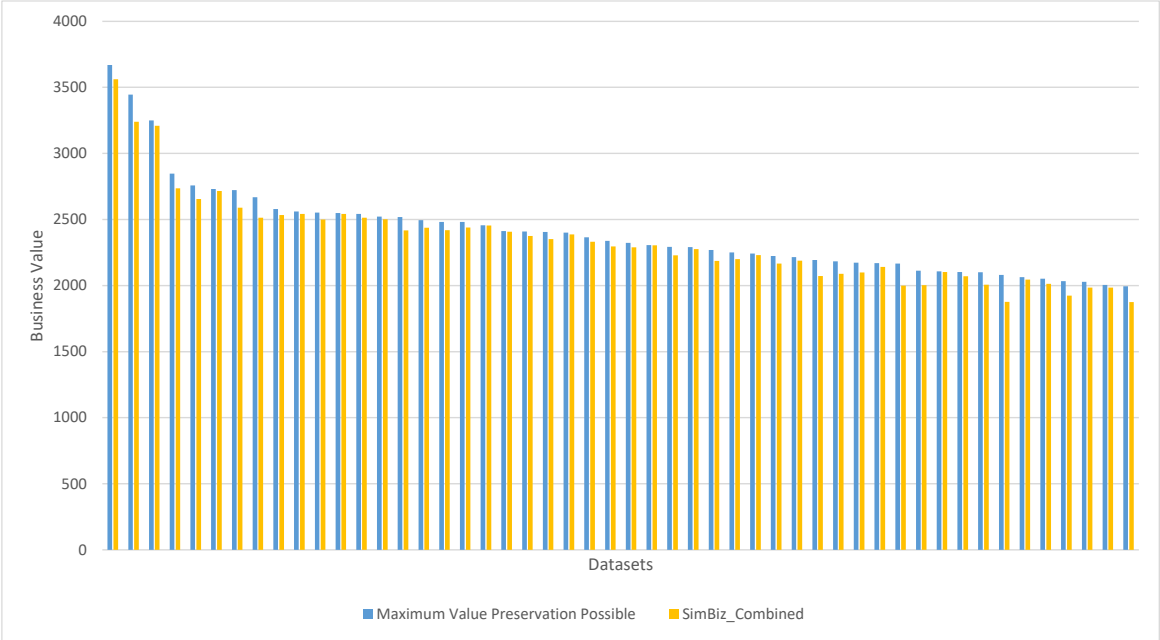


Figure 34: Maximum Preservable Business Value vs. SimBiz_Combined-Plot1 (dataset range: 1 - 50).

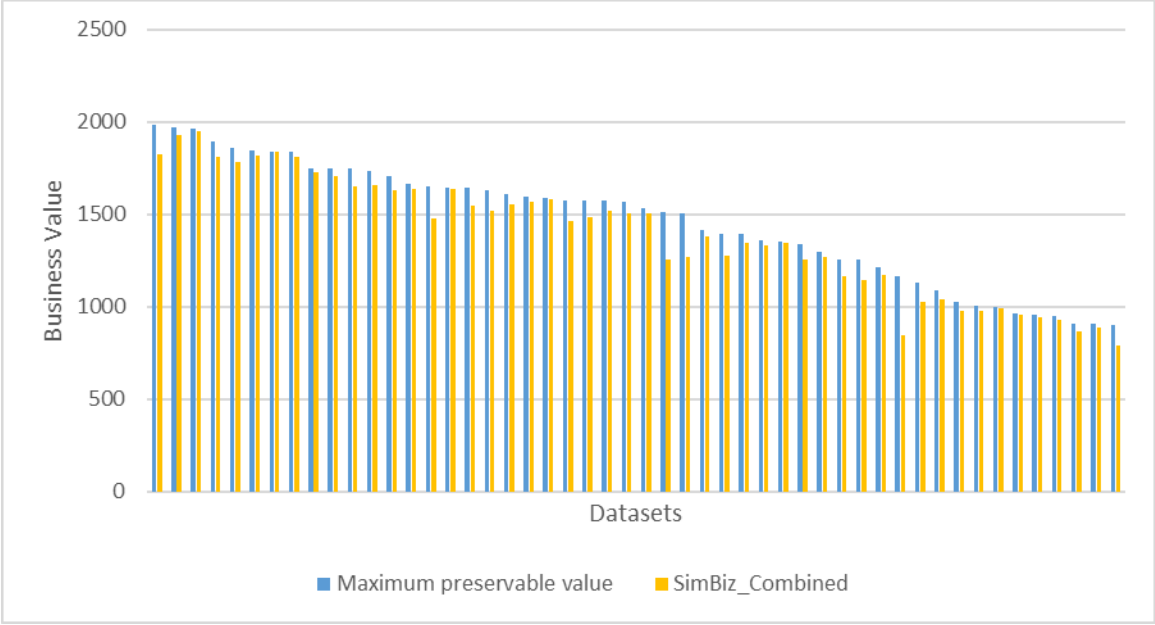


Figure 35: Maximum Preservable Business Value vs. SimBiz_Combined-Plot2 (dataset range: 51 - 100).

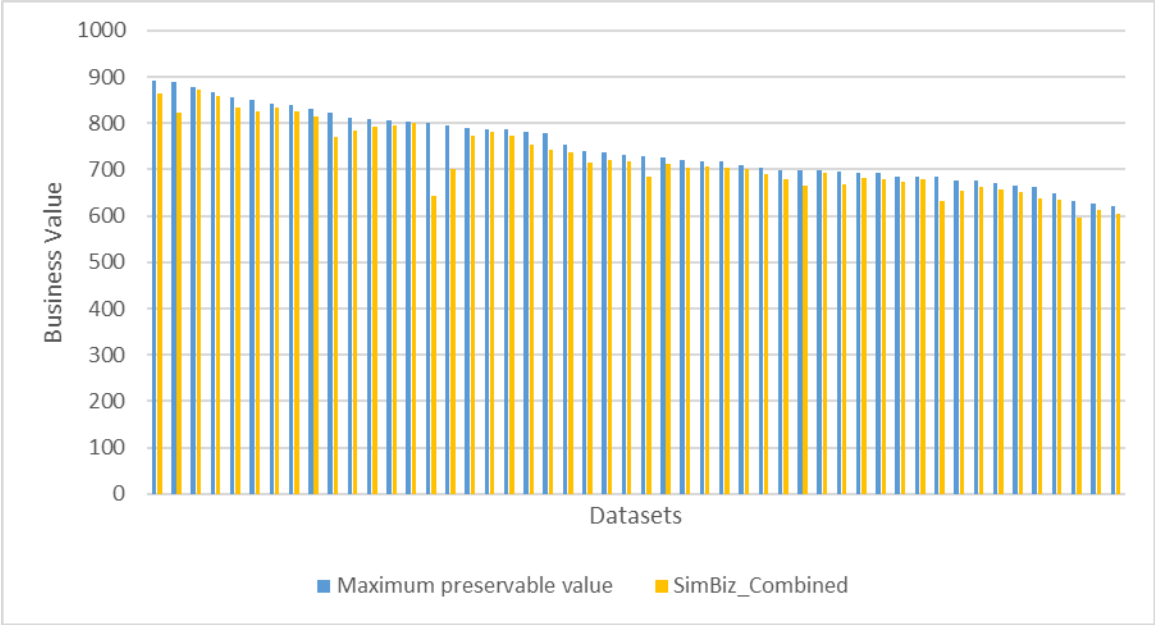


Figure 36: Maximum Preservable Business Value vs. SimBiz_Combined-Plot3 (dataset range: 101 - 150).

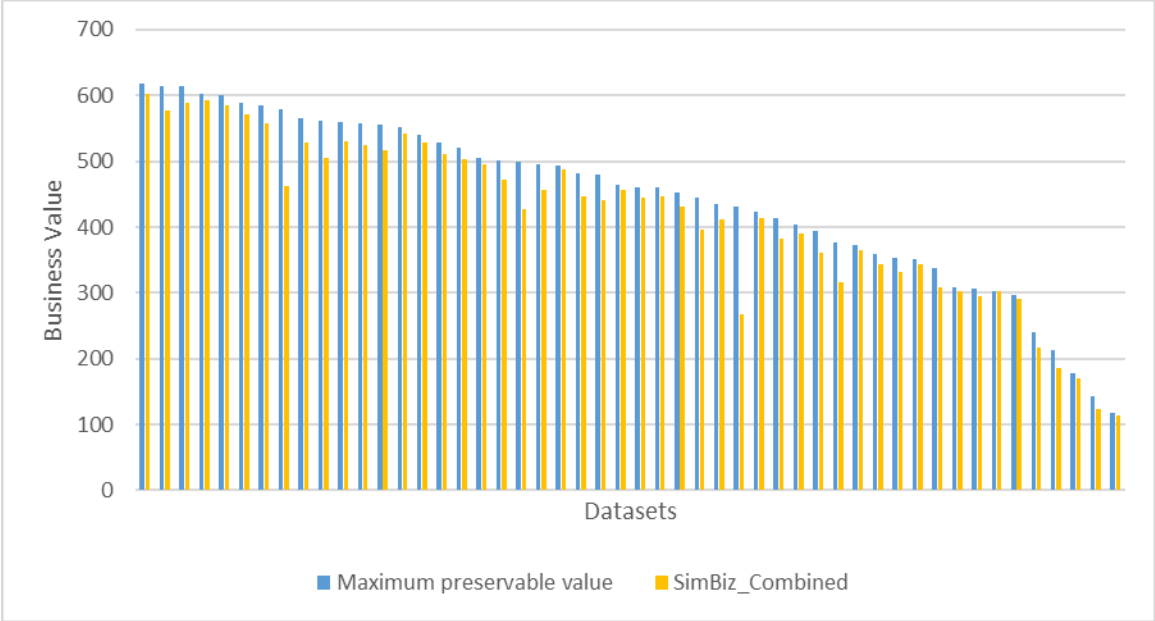


Figure 37: Maximum Preservable Business Value vs. SimBiz_Combined-Plot4 (dataset range: 151 - 200).

Figure 38 to 45 show results of 200 datasets for all configurations:

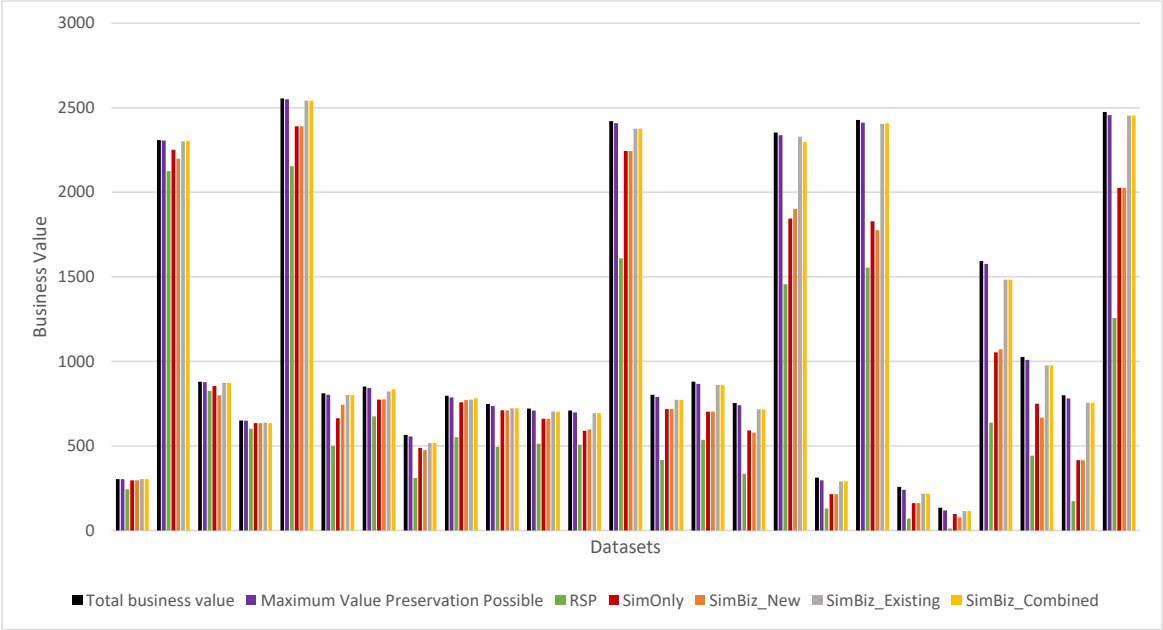


Figure 38. All configurations with respect to Total Business Value and Maximum Preservable Value-Plot1 (dataset range: 1 - 25).

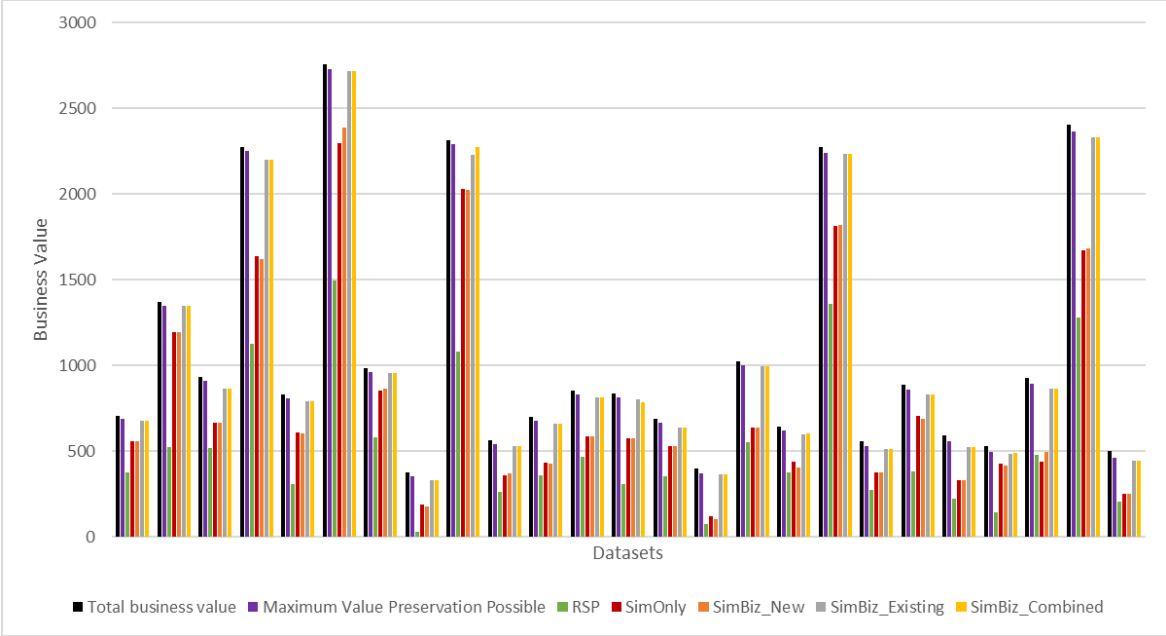


Figure 39. All configurations with respect to Total Business Value and Maximum Preservable Value-Plot2 (dataset range: 26 - 50).

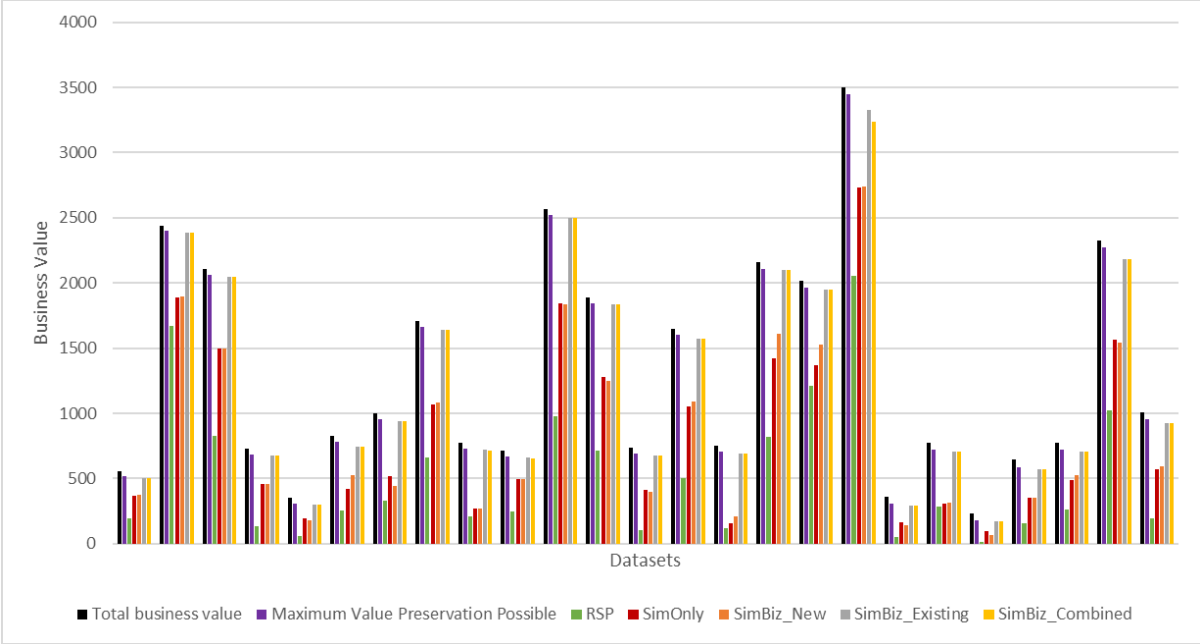


Figure 40. All configurations with respect to Total Business Value and Maximum Preservable Value –Plot3 (dataset range: 51 - 75).

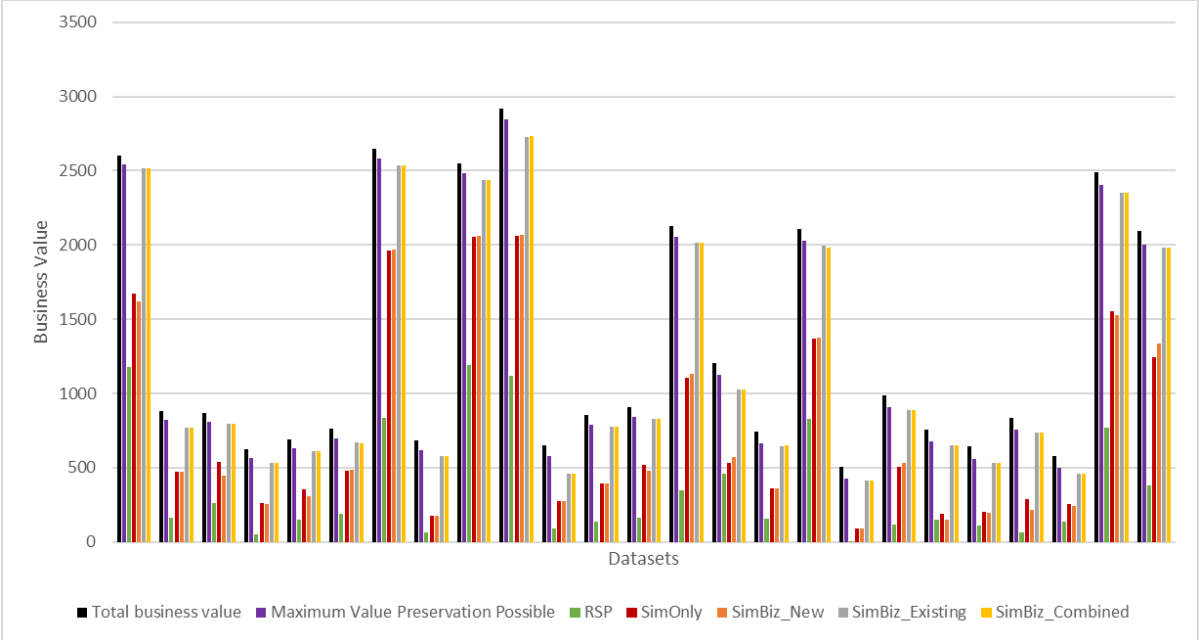


Figure 41. All configurations with respect to Total Business Value and Maximum Preservable Value –Plot4 (dataset range: 76 - 100).

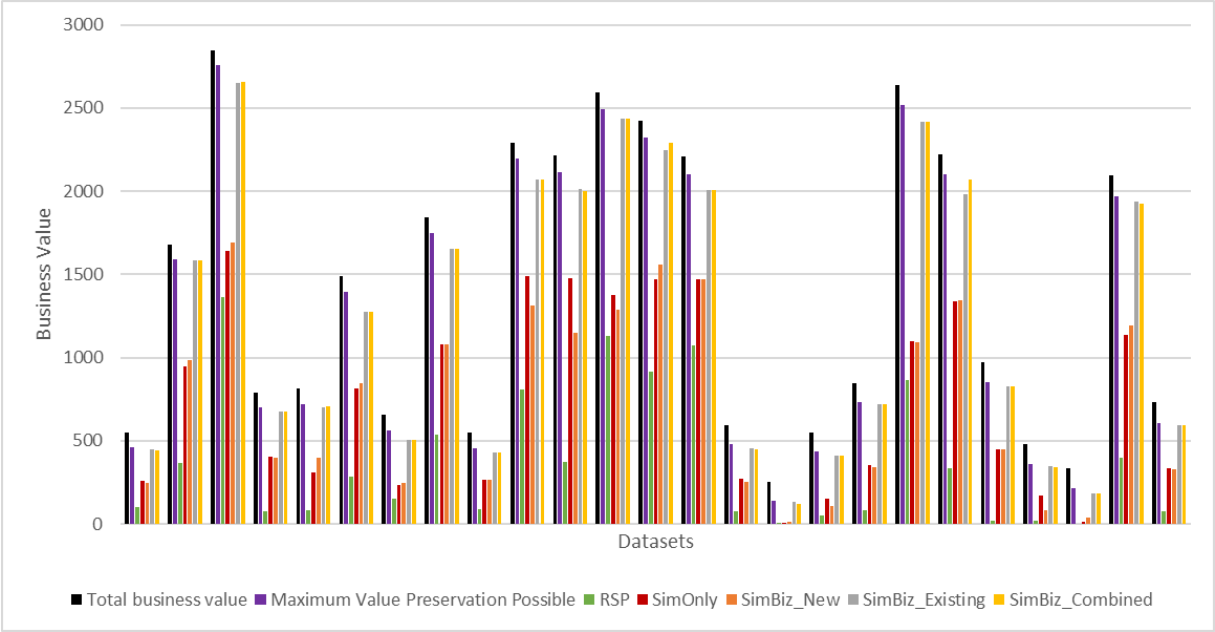


Figure 42. All configurations with respect to Total Business Value and Maximum Preservable Value –Plot5 (dataset range: 101 - 125).

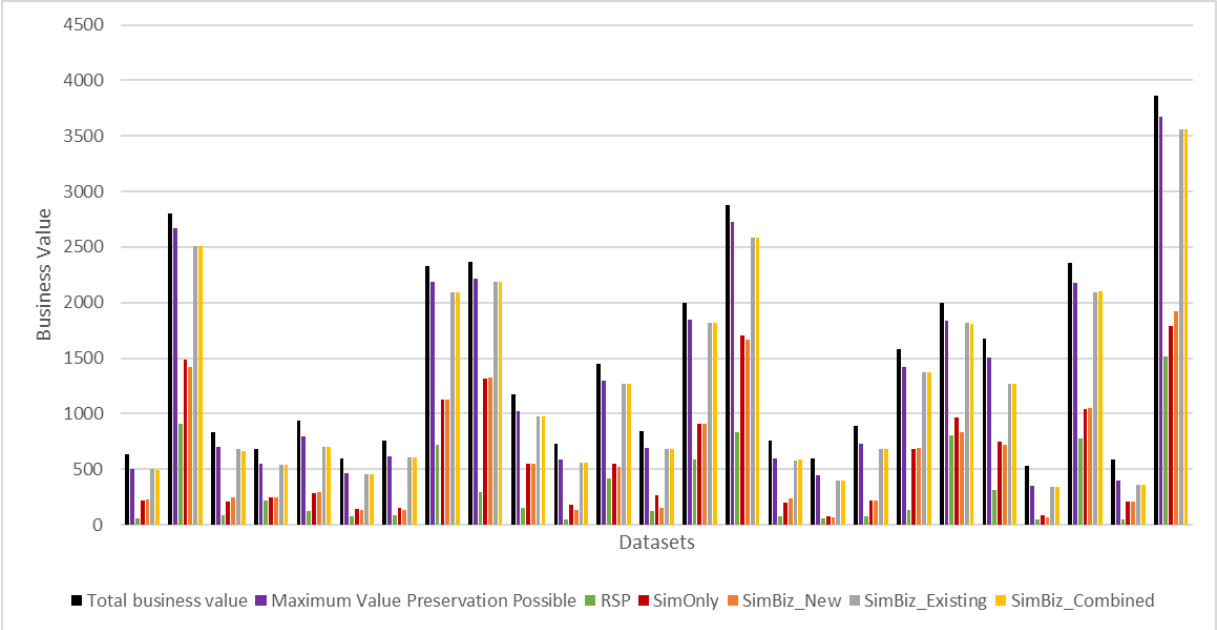


Figure 43. All configurations with respect to Total Business Value and Maximum Preservable Value –Plot6 (dataset range: 126 - 150).

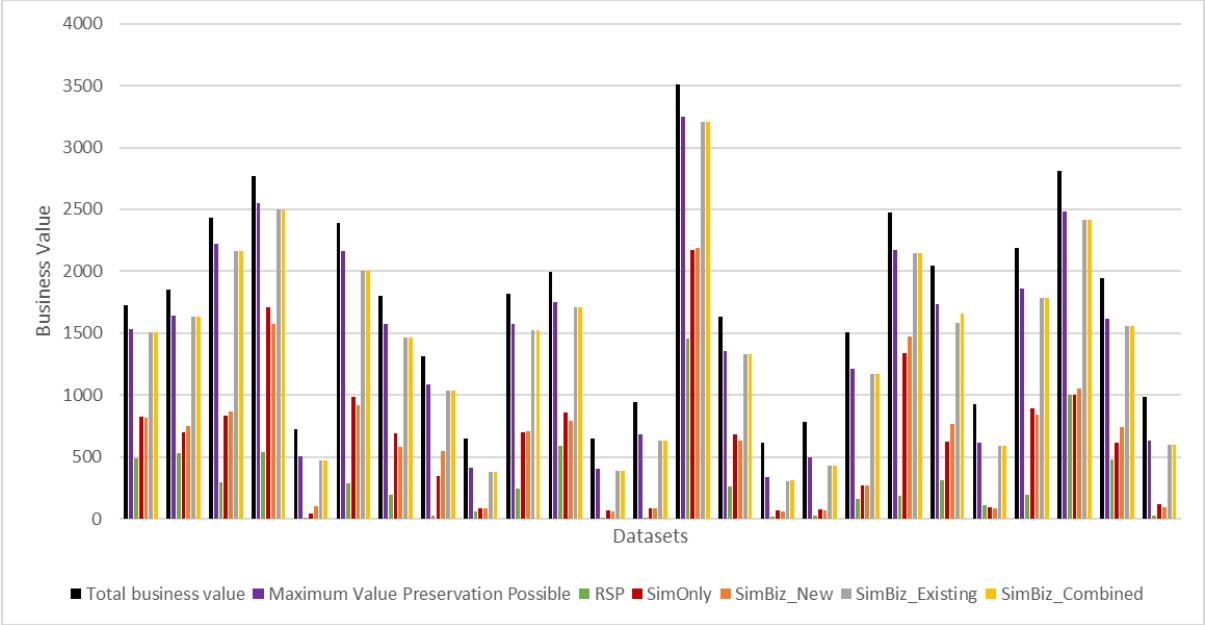


Figure 44. All configurations with respect to Total Business Value and Maximum Preservable Value –Plot7 (dataset range: 151 - 175).

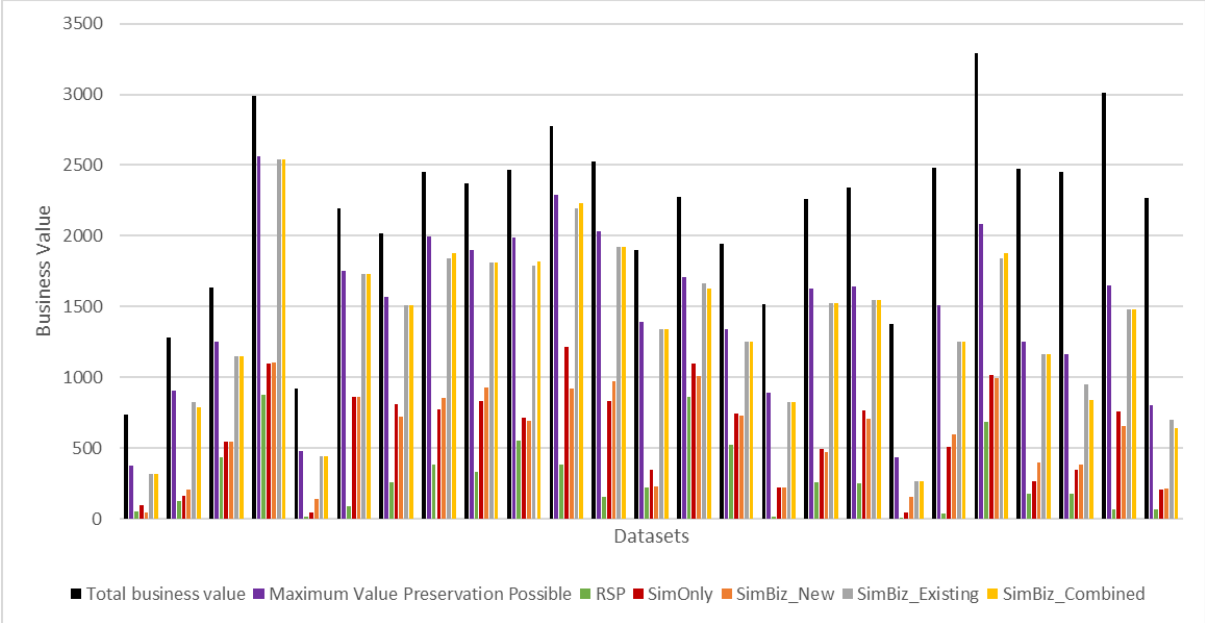


Figure 45. All configurations with respect to Total Business Value and Maximum Preservable Value –Plot8 (dataset range: 176 - 200).

Figure 46 to 53 show results of 200 datasets for all configurations in terms of Percentage of Value Preserved to the Maximum Preservable Value:

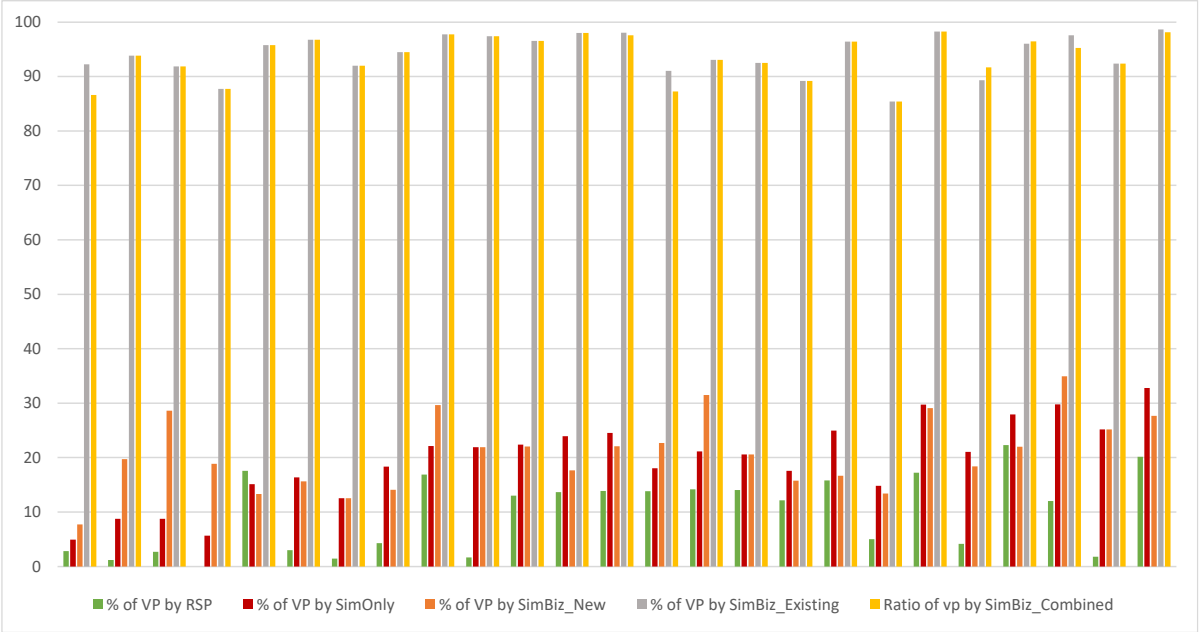


Figure 46. Percentage of Value Preserved by all configurations-Plot1 (dataset range: 1 - 25).

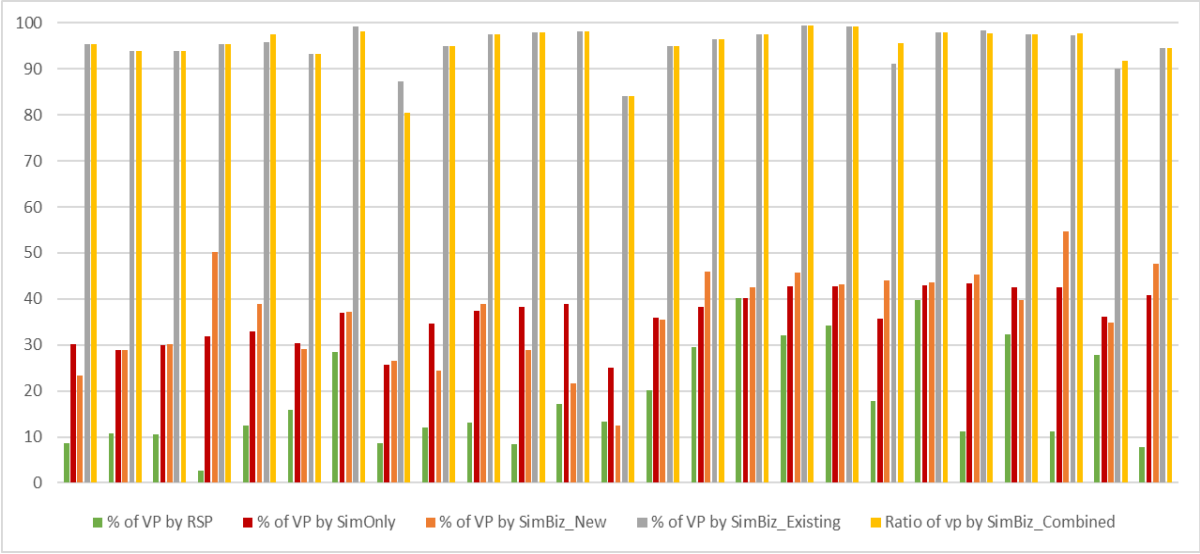


Figure 47. Percentage of Value Preserved by all configurations-Plot2 (dataset range: 26 - 50).

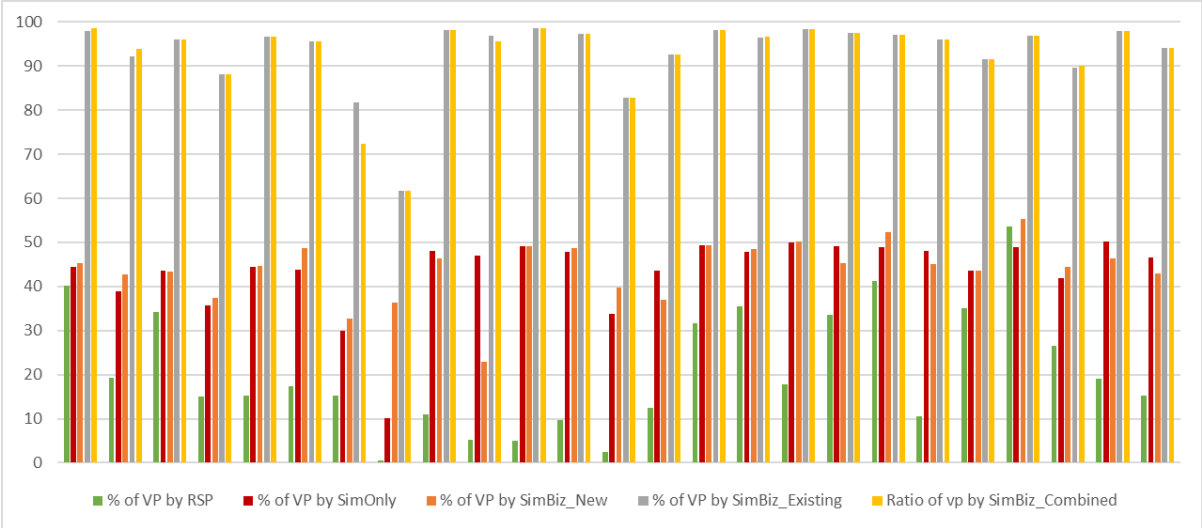


Figure 48. Percentage of Value Preserved by all configurations-Plot3 (dataset range: 51 - 75).

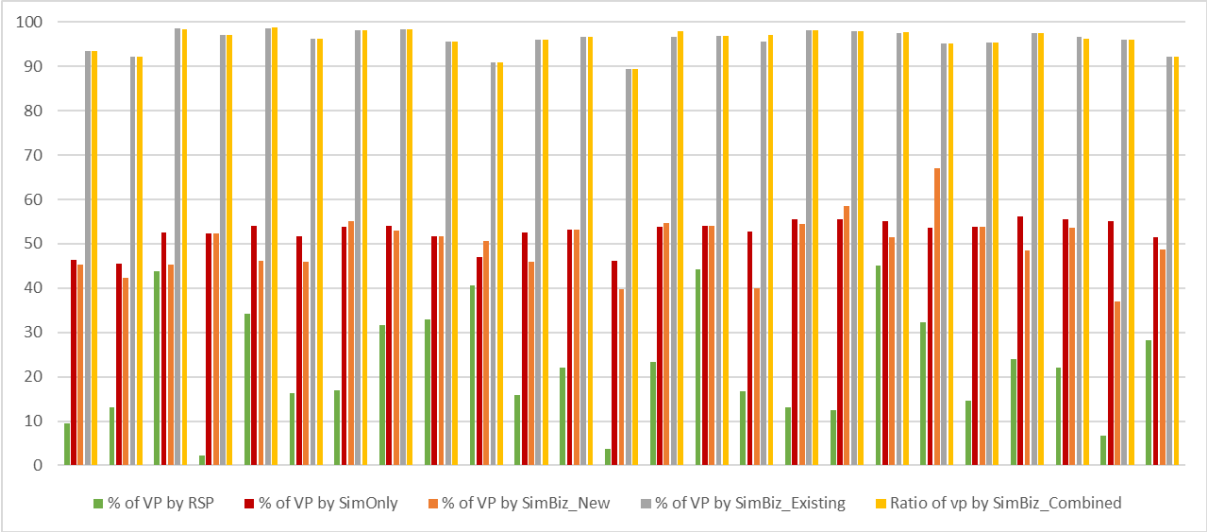


Figure 49. Percentage of Value Preserved by all configurations-Plot4 (dataset range: 76 - 100).

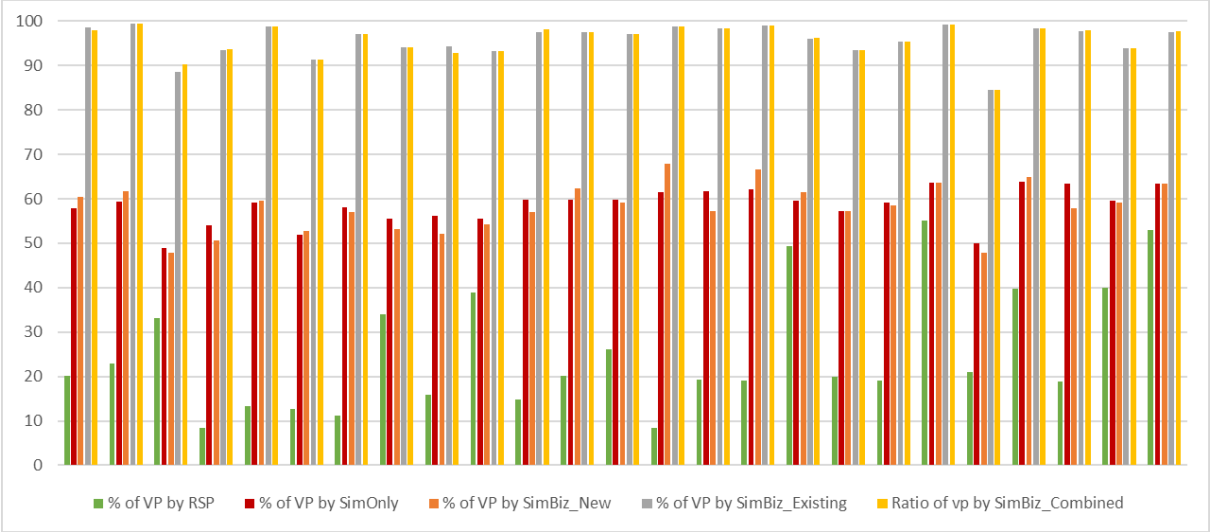


Figure 50. Percentage of Value Preserved by all configurations-Plot5 (dataset range: 101 - 125).


Figure 51. Percentage of Value Preserved by all configurations-Plot6 (dataset range: 126 - 150).


Figure 52. Percentage of Value Preserved by all configurations-Plot7
(dataset range: 151 - 175).

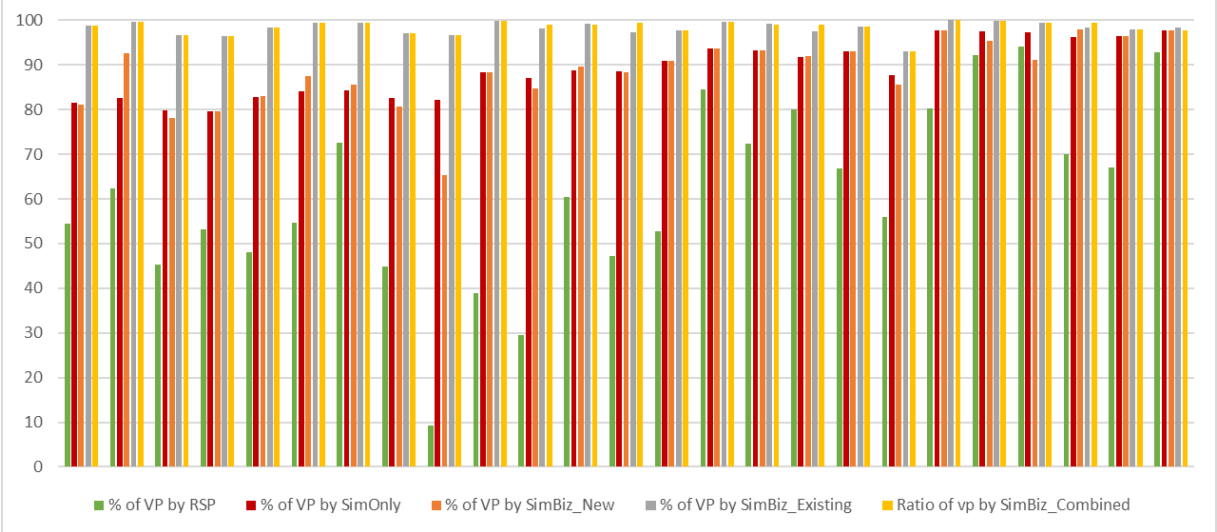


Figure 53. Percentage of Value Preserved by all configurations-Plot8

(dataset range: 176 - 200).