

FLUMINENSE FEDERAL UNIVERSITY

Isela Mendoza Del Castillo

Combining Verification and Validation Methods to Cover Software Quality Characteristics

Dissertation presented to the Computing Graduate program of the Fluminense Federal University in partial fulfilment of the requirements for the degree of Master of Science. Research area: Systems and Information Engineering.

Advisor:
Uéverton dos Santos Souza

NITERÓI

2018

Ficha catalográfica automática - SDC/BEE

M539c Mendoza del Castillo, Isela
 Combining Verification and Validation Methods to Cover
 Software Quality Characteristics / Isela Mendoza del Castillo;
 Uéverton dos Santos Souza, orientador. Niterói, 2018.
 82 p. : il.

 Dissertação (mestrado)-Universidade Federal Fluminense,
 Niterói, 2018.

DOI: <http://dx.doi.org/10.22409/PGC.2018.m.07346482190>

 1. Software Quality Characteristics. 2. Combining
 Verification and Validation Methods. 3. Parameterized
 Complexity. 4. Algorithm fixed-parameter tractable (FPT-
 algorithm). 5. Produção intelectual. I. Título II.
 Souza, Uéverton dos Santos , orientador. III. Universidade
 Federal Fluminense. Escola de Engenharia.

CDD -

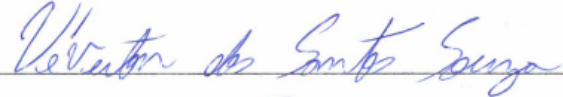
Isela Mendoza Del Castillo

Combining Verification and Validation Methods to Cover Software Quality
Characteristics

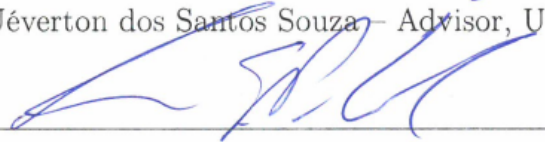
Dissertation presented to the Computing
Graduate program of the Fluminense Federal
University in partial fulfilment of the require-
ments for the degree of Master of Science.
Research area: Systems and Information En-
gineering.

Approved in July of 2018

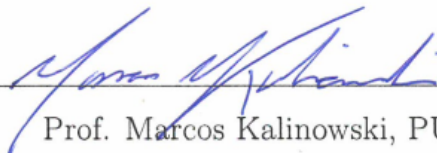
EXAMINATION COMMISSION



Prof. Uéverton dos Santos Souza – Advisor, UFF



Prof. Leonardo Gresta Paulino Murta, UFF



Prof. Marcos Kalinowski, PUC-RJ



Prof. Carlos Vinícius Gomes Costa Lima, UFMG

Niterói

2018

Dedicated to God and my family, my pillars.

Acknowledgments

I would like to thank God for always giving me the strength I need to continue and never give up.

I want to thank infinitely the Computer Institute of the Fluminense Federal University - UFF, in which I studied. Thanks to all my teachers, especially to my advisors: Prof. Uéverton dos Santos Souza and Prof. Marcos Kalinowski. Thanks to all those who collaborated in the research, thanks to the members of the examining commission and thanks to the secretary and security staff of the institute for their cordiality, especially Teresa. Thank to CAPES, CNPq and FAPERJ organizations for the financial support.

I also want to thank my parents Anabel and Jorge Enrique for their support and infinite love. Thank to my husband Ruben for always being by my side, and his family. Thank my little daughter Anna who is my biggest motivation. Thanks to all my Cuban family.

Many thanks also to those I consider as my Brazilian parents: Father Henrique and the presbyter Tatiana of our holy Orthodox church, for all her love and dedication to all of us.

I would also like to thank all my Cuban friends who have been like brothers, with whom I studied here: Jose Angel, Ruslan, Moreno, Jose Ramon, Alberto, Egberto, Elio, Alfredo, Daysel, Angel y Javier. Thanks to my best friends Yanexis, Rosana, Debora, Yenisei and Ely for all my support, care and love with my daughter whenever I needed it. Thanks to Marielena and Dalia, coworkers and friends, for their good recommendations and having trusted me.

Thanks to all my Brazilian friends and wonderful neighbors especially Rose, thank you Brazil for the opportunity and for welcoming us.

“Being educated is the only way to be free”

José Martí

Abstract

Studies point out that a large amount of the total cost of software development concerns quality control purposes. According to the literature, an adequate combination of verification and validation (V&V) methods is important to improve software quality control throughout the development process and to reduce such costs. However there is no concrete evidence on what V&V methods are used to cover each of the software quality characteristics. For this, a survey was applied to experts of the area, obtaining an initial configuration of the relationship between V&V methods and software quality characteristics.

On the other hand, finding an appropriate set of V&V methods that together properly cover the desired quality characteristics of a given project is a NP-hard problem. In this work, is presented a novel approach that combines V&V methods efficiently in order to properly cover a set of quality characteristics. To modelated the problem is using a bipartite graph representing the relationships between V&V methods and quality characteristics. Then, the problem is interpreted as the Set Cover Problem. Although Set Cover is considered hard to be solved, through the theoretical framework of Parameterized Complexity is proposed an FPT-Algorithm (fixed-parameter tractable algorithm) that effectively solves the problem, considering the number of quality characteristics to be covered as a parameter.

Is concluded that the results of the survey are essential for the identification of “best” V&V methods that cover a set of quality characteristics, this initial configurations can be refined according to the cost of the methods, type of project and context to be applied in the industry. The algorithm proposed represents a powerful tool to combine V&V methods optimally, being more scalable and efficient than others like brute force or exhaustive searches in terms of the maintainability and expertise, which represent a valuable contribution to the community.

Keywords: combination, verification, validation, method, software, quality characteristics.

Resumo

Estudos apontam que uma grande parte do custo total do desenvolvimento de software se refere a propósitos de controle de qualidade. De acordo com a literatura, uma combinação adequada de métodos de verificação e validação (V&V) é importante para melhorar o controle de qualidade de software durante todo o processo de desenvolvimento, reduzindo tais custos. No entanto, não há evidências concretas sobre quais métodos de V&V são mais adequados para cobrir cada uma das características da qualidade do software. Para isso, uma pesquisa foi aplicada a especialistas da área, obtendo-se uma configuração inicial da relação entre métodos de V&V e características de qualidade de software.

Por outro lado, encontrar um conjunto apropriado de métodos V&V que juntos cobrem adequadamente as características de qualidade desejadas de um determinado projeto é um problema difícil. Neste trabalho, apresenta-se uma nova abordagem que combina os métodos V&V de forma eficiente para cobrir adequadamente um conjunto de características de qualidade. Para modelar o problema utilizam-se um grafo bipartido que representa as relações entre os métodos V&V e as características de qualidade. Então, o problema é interpretado como o Problema da Cobertura por Conjunto (Set Cover). Embora Set Cover seja considerado difícil de ser resolvido, através do referencial teórico da Complexidade Parametrizada propomos um Algoritmo FPT (algoritmo tratável por parâmetros fixos) que efetivamente resolve o problema, considerando o número de características de qualidade a serem cobertas como parâmetro.

Conclui-se que os resultados da pesquisa são essenciais para a identificação dos “melhores” métodos de V&V que cobrem um conjunto de características de qualidade, estas configurações iniciais podem ser refinadas de acordo com o custo dos métodos, tipo de projeto e contexto a ser aplicado na indústria. O algoritmo proposto representa uma ferramenta poderosa para combinar os métodos V&V de forma otimizada, sendo mais escalável e eficiente do que outros de força bruta em termos de capacidade de manutenção e especialização, representando uma contribuição valiosa para a comunidade.

Palavras-chave: combinação, verificação, validação, método, software, características de qualidade.

List of Figures

2.1	Characteristics of ISO 9126 and ISO 25010	11
2.2	Classification of V&V methods	13
3.1	Studies selection process	27
3.2	Combinations methods type	29
3.3	Combinations categories	29
3.4	Research type	30
3.5	Number of papers by combination methods types	31
3.6	Number of papers by years of publication	31
3.7	Number of papers by publisher and source type	31
4.1	Number of responses by country	37
4.2	Degree of concordance (MAD) between the experts	38
4.3	Results of the first question	40
5.1	Bipartite graph of V&V methods vs. Quality characteristics	45
5.2	Set cover FPT–Algorithm	47
5.3	Algorithm execution. State of the graph after the preprocessing step	48
5.4	Algorithm execution. State of the graph after the recursive call	49
5.5	Optimal solution	49

List of Tables

3.1	Search String	21
3.2	Set of Initial Papers	24
3.3	First Iteration	24
3.4	Results of First Iteration	25
3.5	Second Iteration	26
3.6	Results of Second Iteration	26
3.7	Third Iteration	26
4.1	Threats and Treatment	36
4.2	Survey Result	39
5.1	Experiment Results	53

Abbreviations and Acronyms List

V&V: Validation and Verification.

ISO : International Organization for Standardization.

SM: Systematic Mapping

SCP: Set Cover Problem.

NP: Nondeterministic Polynomial Time.

FPT: Fixed Parameter Tractable.

PR: Peer Review.

WM: Workflow Models.

FSM: Finite-State Machines.

OP: Operational Profile.

MT: Mutation Testing.

ET: Exploratory Testing.

FS: Functional Suitability.

PE: Performance Efficiency.

C: Compatibility.

U: Usability.

R: Reliability.

S: Security.

M: Maintainability.

P: Portability.

Contents

1	Introduction	1
1.1	Context	1
1.2	Background	1
1.3	Problem	3
1.4	Motivation	3
1.5	Goal	4
1.5.1	Specific Objectives	5
1.6	Main Idea	5
1.7	Methodology	6
1.8	Structure Dissertation	7
2	Software Quality and V&V Methods	8
2.1	Introduction	8
2.2	Terms and Definitions	9
2.3	The ISO Standards	10
2.3.1	ISO/IEC 9126:1991	10
2.3.2	ISO/IEC 25010:2011	10
2.4	Description of ISO 25010 Quality Characteristics	11
2.5	Characterization of V&V Methods	12
2.5.1	Based on Intuition and Experience	12
2.5.2	Input Domain-Based Techniques	13
2.5.3	Code-Based Techniques	13

2.5.4	Fault-Based Techniques	14
2.5.5	Usage-Based Techniques	14
2.5.6	Model-Based Testing Techniques	14
2.5.7	Reviews	14
2.6	Conclusions	15
3	Systematic Mapping on Combinations of V&V Methods	16
3.1	Introduction	16
3.2	Systematic Mapping	17
3.3	Protocol Planning	17
3.3.1	Research Questions	17
3.3.2	Search Strategy	19
3.3.2.1	The PICO Criteria and Keywords	19
3.3.2.2	The Controls Papers	19
3.3.2.3	Data Search and Snowballing Strategy	20
3.3.2.4	Data Sources	21
3.3.3	Selection Strategy	21
3.3.3.1	Inclusion Criteria	21
3.3.3.2	Exclusion Criteria	22
3.3.3.3	Procedures of Selecting Studies by Filters	22
3.4	Results	27
3.4.1	Data Extraction Strategy	27
3.4.2	Data Synthesis Strategy	30
3.5	Conclusions	32
4	Relating V&V Methods to Software Product Quality Characteristics: Results of an Expert Survey	33
4.1	Introduction	33

4.2	Survey Planning	34
4.2.1	Main Goal and Scope	34
4.2.2	Population	34
4.2.3	Survey Questions	34
4.2.4	Metrics	35
4.2.5	Execution Strategy	35
4.2.6	Statistical Techniques	35
4.2.7	Instrumentation	36
4.2.8	Validity Assessment	36
4.3	Operation	37
4.4	Survey Results	38
4.4.1	Answering the Research Question 1	40
4.4.2	Answering the Research Question 2	41
4.5	Conclusions	41
5	An Efficient Algorithm for Combining V&V Methods	42
5.1	Introduction	42
5.2	Parameterized Complexity	42
5.2.1	Fixed-Parameter Tractable (FPT) Approach	43
5.2.2	Scalability of the FPT-algorithms	44
5.3	Modeling the Problem	44
5.4	FPT-Algorithm to Combine V&V Methods	46
5.4.1	Execution of the Set Cover Algorithm	48
5.4.2	Running Time Analysis	50
5.5	Computational Experiments	51
5.5.1	Integer Linear Programming Model	51
5.5.2	Results	52

Contents	xiv
5.6 Discussion	54
5.7 Conclusions	54
6 Concluding Remarks	56
6.1 Conclusions	56
6.2 Contributions	57
6.3 Future Work	57
References	58
Appendix A – Expert survey: V&V Methods and Quality Attributes.	65

Chapter 1

Introduction

1.1 Context

Studies suggest high costs related to quality assurance activities in software development projects [61]. The appropriate combination of verification and validation (V&V) methods is seen in the literature as a way to reduce these costs and increase product quality [10]. Over the years, some knowledge has been generated regarding V&V methods when observed in isolation. However, the selection of different V&V methods as well as the interdependencies among them are still not well-understood [31].

Quoting from [37] *“The purpose of V&V is to help the development organization build quality into the system during the life cycle. V&V processes provide an objective assessment of products and processes throughout the life cycle. This assessment demonstrates whether the requirements are correct, complete, accurate, consistent, and testable. The V&V processes determine whether the development products of a given activity conform to the requirements of that activity and whether the product satisfies its intended use and user needs.”*

1.2 Background

The quality of software products is strongly dependent on the appropriate combination of V&V methods employed during development [31]. Experimental studies have long demonstrated that the use of combinations of different V&V methods to ensure the quality of a software is more effective than using isolated methods [10].

Elbertzhager et al. [27] conducted a mapping study concerning the combination of

V&V methods. They describe two fundamental approaches: Compilation and Integration. We focus on the Compilation approach since our purpose is purely to combine existing V&V methods (Compilation process). We are not focusing on creating new techniques by combining different methods in one, nor in using the results of the application of some technique as an instance to apply another one (Integration of V&V methods).

In order to establish how other works perform the combination of V&V methods in the Compilation approach, Elberzhager et al. [27] created a categorization to classify and organize these studies into three subgroups. In the first subgroup, static and dynamic techniques are combined, focusing on thread escape analysis, atomicity analysis, protocol analysis, vulnerability analysis, concurrent program analysis or on defects in general. All these combinations are supported by open-source or proprietary tools. The second subgroup compares different testing and inspection techniques discussing advantages and disadvantages among them. In most cases, two or three techniques are compared to each other. Several studies initially perform inspections, followed by some tests, corroborating then the effectiveness of the combination of both techniques. The last subgroup describes other combinations, such as testing techniques and inspections combined with formal specifications, bug-finding tools, comprehensive quality control processes in industrial environments, comprising several inspections and technical tests, requirements and static analysis, tutorials, simulations, and vision-based approaches.

The most cited papers in the SM [27] regarding the Compilation approach are: Basili [7], Kamsties and Lott [48], and Wagner et al. [75]. Basili [7] makes a comparison of three software testing techniques: reading of code by gradual abstraction, functional testing using equivalence partition and border value analysis, and structural testing using total coverage of criticism, according to efficiency, cost, and fault detection classes. Kamsties and Lott [48], evaluate three techniques through a controlled experiment: reading of code by gradual abstraction, functional (black-box) testing and (3) structural (white-box) testing. Wagner [75] describes a case study where several projects are analyzed in an industrial environment. In this project, automatic static analysis, testing, and reviews are used to detect defects. Their results show that these techniques complement each other and that they should be combined.

In the SM [27], papers were analyzed until 2010. This led us to carry out an update regarding the compilation approach, with the aim of finding more relevant and recent papers from 2010 to present.

Dwyer and Elbaum [26] suggest an approach based on dividing V&V methods in two

main classes: those that make dynamic analyses (or focused on behavior of the system, e.g., testing) and those that use static analysis (typically focused on a single property of the system at a time). Runeson et al. [67] compare code inspections and structural unit tests by analyzing three replications of an experiment in order to know which method finds more faults. Olorisade et al. [63] investigate the effectiveness of two tests techniques (partition of equivalence class and decision coverage) and one review technique (code by abstraction) in terms of their ability to detect faults. Cotroneo et al. [22] combine testing techniques adaptively, based on machine learning, during the testing process, by learning from past experience and adapting the technique selection to the current testing session. Bishop et al. [9] combine a monotonicity analysis with a defined set of tests, showing that, unlike “independent” dynamic methods, this combination provides a full error coverage. Solari and Matalonga [70] study the behavior of two techniques, equivalence partition and decision coverage, to determine the types of defects that are undetectable for either of them. Finally, Gleirscher et al. [34] analyze three different techniques of automated static analysis: code clone detection, bug pattern detection, and architecture conformance analysis. They claim that this combination tends to be affordable in terms of application effort and cost to correct defects.

1.3 Problem

A significant part of the software industry is made up of small and medium-sized companies that, given the lack of guidelines for performing the right combination of V&V methods, have difficulties in optimizing this combination for their context, increasing the costs of resources and time and mainly harming the quality of the produced software.

1.4 Motivation

It is known that an adequate combination of V&V methods outperforms any method alone [29]. Performing this combination properly is beneficial to ensure product quality and maximize the chances of success of software development projects. In the literature cited above does not provide objective subsidies that support the organization in deciding which V&V methods cover which quality characteristics (e.g., considering the quality characteristics described in the ISO 25010 quality model standard). This is a problem complex and not trivial that depends on several factors as costs, context and type of project.

Taking into account the quality characteristics of the ISO 25010 standard [41] and series of V&V methods compiled mainly from the SWEBOK [12] and other sources [74] [76], this work suggests an initial configuration of the relationship between which V&V methods are the most appropriate to cover each of the ISO 25010 characteristics. This proposal is presented at a high level, in a general way that will need to be refined in detail to be implemented in the industry. This configuration represents an important starting point for the solution of the problem in question.

Finding a set of methods that together properly cover all quality characteristics of interest can be seen as a Set Cover Problem (SCP) [49]. The SCP is a classic NP-hard problem in the computational complexity area, whose decision version belongs to the list of the 21 Karp's NP-complete problems [49]. This means that when the number of methods or quality characteristics increase, the performance of the algorithm drastically decreases.

The existence of efficient algorithms to solve NP-complete, or otherwise NP-hard, problems is unlikely, if the input parameters are not fixed; all known algorithms that solve these problems require exponential time (or at least superpolynomial time) in terms of the input size. However, some problems can be solved by algorithms for which we can split the running time into two parts: one exponential, but only with respect to the size of a fixed parameter, and another polynomial in the size of the input. Such algorithms are denoted FPT (fixed-parameter tractable) in the Parameterized Complexity field, because the problem can be solved efficiently for small values of the fixed parameter [23] [33] [62]. This field emerged as a promising alternative for working with NP-hard problems [68].

To obtain the optimal combination of methods in reasonable computational time an algorithm is proposed, adopting a parameterized approach, that consider the set of quality characteristics as the fixed parameter, and obtaining an algorithm classified as FPT.

1.5 Goal

The main goal of this work is suggest an approach to obtain the optimal combination of V&V methods that cover the quality characteristics of ISO 25010 standard, the smallest number of methods covering all the characteristics.

1.5.1 Specific Objectives

To satisfy the main goal, the following specific objectives are proposed:

- Classify and analyze the existing bibliography about the combination of V&V methods executing a secondary study, a systematic mapping (SM).
- Identify, classify and characterize an initial set of V&V methods.
- Plan and execute a survey to determine which method (s) meets which ISO 25010 quality characteristics.
- Model the problem in question.
- Implement an efficient algorithm to obtain the optimal combination of the least amount of V&V methods that cover all the ISO 25010 characteristics.

1.6 Main Idea

With the purpose of obtaining concrete evidence on which V&V methods are most suitable to address each of the ISO 25010 characteristics, as a first instance, a survey is conducted. Experts with PhD in Software and Quality Engineering and relevant publications in the area were interviewed to know their opinion and obtain, in a general way, an initial configuration of the relationship between the V&V methods and the software quality characteristics. The set of V&V methods, and their classification, used in the survey and in the approach to get the optimal combination is extracted from SWEBOK [12] and other sources [74] [76], a reliable source and one of the most representative software engineering books.

The relation between the characteristics and the methods can be modeled as an undirected bipartite graph. The problem of finding the smallest combination of methods that cover all the ISO quality characteristics is interpreted as SCP. The set of characteristics is the universe U , and each method defines M_i , a subset of U that is covered by that method. We need to find a smallest set of subsets that cover U . The available data (instance) has exactly 8 characteristics and 20 methods. At most, they can be close to 50 methods in the literature.

The SCP is a classic NP-hard problem in the computational complexity area, whose decision version belongs to the list of the 21 Karp's NP-complete problems. The existence

of efficient algorithms to solve NP-complete, or otherwise NP-hard, problems is unlikely, if the input parameters are not fixed; all known algorithms that solve these problems require exponential time (or at least superpolynomial time) in the total size of the input.

Nevertheless, if some parameter k of the problem is set at a small value or the growth of k is relatively small, then this type of problem can still be considered “manageable” despite its traditional classification as “intractable”.

To obtain an optimal solution, parameterize the problem by the characteristic to be covered by the V&V methods through an algorithm classified as FPT (fixed parameter tractable) in the Theory of Parameterized Complexity it is proposed. This type of algorithms emerged as a promising alternative to work with NP-hard problem, in this way the problem can be solved efficiently considering the small values of a fixed parameter.

The goal of the exact algorithm is to obtain the optimal combination (least amount) of V&V methods that cover all the quality characteristics. The objective function is the number of methods that cover all the characteristics. The parameter to be set is the number of quality characteristics of ISO 25010, parameterizing the problem of the cover of the set by the number of characteristics that will cover the V&V methods.

The set of methods presented and their relationship with the software quality characteristics obtained from the survey, represent only the basis to be refined and applied to the industry. To refine the initial set, in addition to increase or modify the set of V&V methods to consideration, each organization have to take into account other important elements such as: the cost of application of each V&V method for each of the software quality characteristics, the context and type of project in which the approach will be applied. The step of refinement is suggested for future work. The present study focuses on the approach to obtain an optimal combination on any data set that is modeled as a SCP through an FPT-algorithm.

1.7 Methodology

The methodology will follow an experimental approach to developing software technologies with support from primary and secondary studies. This approach extends the Shull methodology [69] by including surveys and secondary studies and has already been successfully adopted in other contexts for example by Kalinowski et al. in [46] [47]. Primary and secondary studies will be planned and conducted to support the achievement of the proposed objectives as described in the next section: Structure Dissertation.

1.8 Structure Dissertation

The dissertation is organized in 6 chapters including this Chapter 1 Introduction, followed by Chapter 2 Software Quality and V&V Methods where terms related to software quality are defined, existing ISO Standards are analyzed, quality characteristics of the ISO 25010 Standard are described, and the selected V&V Methods are classified and characterized. In Chapter 3 Systematic Mapping on Combinations of V&V Methods, is conducted for the obtaining and classification of relevant studies on combination approaches of V&V methods is conducted. In Chapter 4 Survey of V&V Methods for ISO 25010 Quality Characteristics, a survey towards the experts of the academy is executed and analyzed, with the purpose of obtaining concrete evidence on the relationship between quality characteristics and V&V methods. In Chapter 5 An Efficient Algorithm for Combining V&V Methods, presents the FPT-Algorithm that obtains the optimal combination, the experiments performed are executed and analyzed. Finally Chapter 6 Concluding Remarks with the contributions and future works.

Chapter 2

Software Quality and V&V Methods

2.1 Introduction

Quality is the set of properties that allow to establish the value of an object. On the other hand, a Software is a set of programs, instructions and computer rules to perform certain tasks on a computer. This leads to the definition of Software Quality, the degree to which a system meets the specified requirements and expectations of a customer or user [74]. For the industry, quality assurance in software development projects has become a high-cost activity. An adequate selection of verification and validation methods (V&V) to ensure that the product is correctly implemented and meets its specifications, is essential for reducing these costs [29] [61].

To guarantee the quality of a product there are standards such as ISO 25010 [41]. The standard specifies the main attributes or characteristics that products should meet to achieve its quality. V&V methods are used to evaluate the quality of some particular software. Unfortunately, the selection of different V&V methods as well as the interdependencies among them are still not well understood. Today the software development industry is faced with the problem of choosing specific V&V methods to evaluate the quality of the software, since an inadequate selection of these methods can generate high costs and a greater effort in software development [29] [61].

This chapter provides a series of state-of-the-art terms and definitions related to software quality, followed by a background on ISO 9126 quality standard and its evolution towards ISO 25010. As well as a brief description of each of the ISO 25010 quality standard characteristics, and of each selected V&V methods. This description is used as supporting documentation in the survey form present in Appendix A.

2.2 Terms and Definitions

Software Quality is the degree to which a system, component or process meets specified requirements, the degree to which a set of inherent characteristics fulfils specifications. This concept relies to the ability of a product, service, system, component or process to meet customer or user needs, expectations or requirements [74].

The concepts of Process and Product quality are closely related. According to [39], high quality processes also lead to high quality products. It is important to note that:

Process Quality is made through institutionalization and continuous improvement of the processes that are used for software development. A software quality process is composed by activities, tasks, procedures, etc., used to produce and maintain software [74].

Product Quality is the degree to which the system satisfies the stated and implicit needs of its various stakeholders, and thus provides value. These stakeholders' needs (functionality, performance, security, maintainability, etc.) are represented in the quality model, which categorizes the product quality into characteristics and subcharacteristics [41].

The quality of the processes is important to deliver high quality software products. However, many factors influence the quality of the product itself, so it is necessary to evaluate and monitor the quality directly in the product, and improve the processes that create them [74].

This study focuses on the software product. ISO 25010 standard characteristics are evaluated in the product finding defects in the product in order to guarantee software quality.

The ISO 25010 Standard defines the quality model that is considered the cornerstone of a product quality evaluation system. The quality model determines which quality characteristics will evaluate the properties of a software product [41].

The **Verification** is the evaluation of the correct construction of a product. It expresses whether it meets the specifications and requirements [12] [74].

The **Validation** is the assurance that a product was built and that it fulfills its specific purpose [12]. It is the activity that evaluates if a work result adjusts to the expectations of the stakeholders [74].

Two types of analysis to find defects during V&V process are static analysis and dynamic analysis [74].

Static Analysis is the process of evaluating a system or component according to its form, structure, content or documentation, that is, the evaluation of a software without executing it. Examples of static analysis are reviews and inspections [74].

Dynamic Analysis is the process of evaluating a system or component based on its behavior during execution. It is the opposite of the static analysis. In this case it is necessary to run the software to analyze it. Examples of dynamic analysis are testing techniques [74].

2.3 The ISO Standards

Quoting from [40] “ *International standards make things work. They offer world-class specifications for products, services and systems, to ensure quality, safety and efficiency. They are essential to facilitate international trade. The ISO standards creates documents that provide requirements, specifications, guidelines or features that can be used consistently to ensure that materials, products, processes and services are adequate for their purpose.*”

This section presents, in summary, the evolution of the quality model of ISO standards for a software product.

2.3.1 ISO/IEC 9126:1991

The ISO 9126 is an international standard for the evaluation of software quality created in 1991 designed to help organizations ensure that they meet customer needs and that they comply with regulatory and legal requirements related to the product [13]. Was later replaced by ISO 25000 in 2011 following the same concepts: classify software quality into a structured set of characteristics and subcharacteristics.

2.3.2 ISO/IEC 25010:2011

The product quality model defined by ISO 25010 is composed of eight quality characteristics, which determine the properties of a software product for its evaluation [41]. These standard is an evolution of the ISO/IEC 9126 standard. Note that two more characteris-

tics were added in the ISO 25010 with respect to ISO 9126, see in Figure 2.1, they are: Compatibility and Security. Security, that was one of the subcharacteristics of Functionality following ISO 9126, is now one of the main characteristics in this latest version of the standard. In the next section, these characteristics are described in more detail.

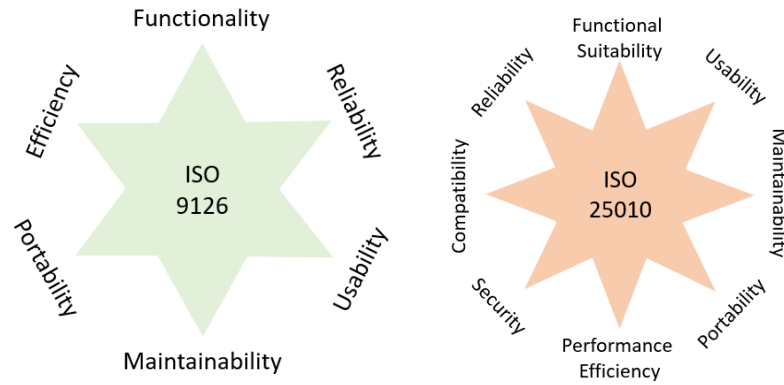


Figure 2.1: Characteristics of ISO 9126 and ISO 25010

2.4 Description of ISO 25010 Quality Characteristics

Next, the software quality characteristics of ISO 25010 and its secondary characteristics will be briefly described [41]:

Functional Suitability: Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. Is composed for the subcharacteristics: Functional Completeness, Functional Correctness and Functional Appropriateness.

Performance efficiency: Represents the performance relative to the amount of resources used under stated conditions. Is composed for the subcharacteristics: Time Behavior, Resource Utilization and Capacity.

Compatibility: Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment. Is composed for the subcharacteristics: Co-existence and Interoperability.

Usability: Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. Is composed for the subcharacteristics: Appropriateness Recognizability, Learnability, Operability, User Error Protection, User Interface Aesthetics and Accessibility.

Reliability: Degree to which a system, product or component performs specified functions under specified conditions for a specified period. Is composed for the subcharacteristics: Maturity, Availability, Fault Tolerance and Recoverability.

Security: Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. Is composed for the subcharacteristics: Confidentiality, Integrity, Non-repudiation, Accountability and Authenticity.

Maintainability: Degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. Is composed for the subcharacteristics: Modularity, Reusability, Analyzability, Modifiability and Testability.

Portability: Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. Is composed for the subcharacteristics: Adaptability, Installability and Replaceability.

2.5 Characterization of V&V Methods

Some software Verification and Validation (V&V) methods and their classification, extracted mainly from the Swebok [12] and the other relevant sources [74] [76], was compiled. The Figura 2.2 show a classification of the selected methods according the literature [12], following the description of each of them in the next subsections.

2.5.1 Based on Intuition and Experience

Ad Hoc: Tests are derived relying on the software engineer's skill, intuition, and experience with similar programs.

Exploratory Testing: Is defined as simultaneous learning, test design, and test execution, that is, the tests are not defined in advance in an established test plan, are dynamically designed, executed, and modified.

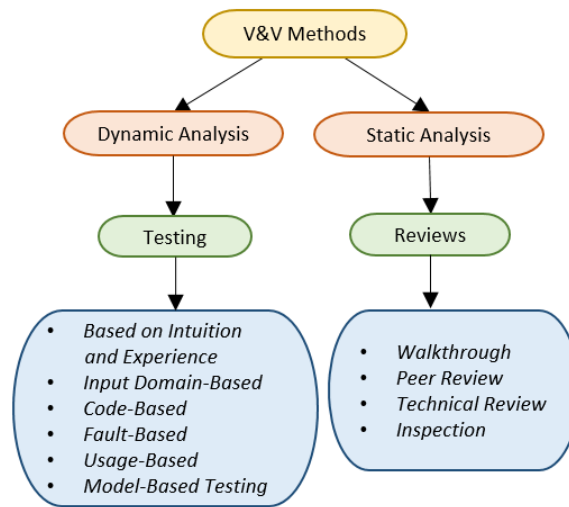


Figure 2.2: Classification of V&V methods

2.5.2 Input Domain-Based Techniques

Equivalence Partitioning: Involves partitioning the input domain into a collection of subsets (or equivalent classes) based on a pacified criterion or relation.

Pairwise Testing: Test cases are derived by combining interesting values for every pair of a set of input variables instead of considering all possible combinations.

Boundary-Value Analysis: Test cases are chosen on or near the boundaries of the input domain of variables, with the underlying rationale that many faults tend to concentrate near the extreme values of inputs.

Random Testing: Tests are generated purely at random. This form of testing falls under the heading of input domain testing since the input domain must be known to be able to pick random points within it.

Cause-Effect Graphing: Represent the logical relationships between conditions (roughly, inputs) and actions (roughly, outputs). Test cases are systematically derived by considering combinations of conditions and their corresponding resultant actions.

2.5.3 Code-Based Techniques

Control Flow-Based Criteria: Are aimed to covering all the statements, blocks of statements, or specified combinations of statements in a program.

Data Flow-Based Criteria: In data flow-based testing, the control flow graph is

annotated with information about how the program variables are defined, used, and killed (undefined).

2.5.4 Fault-Based Techniques

Error Guessing: In error guessing, test cases are specifically designed by software engineers who try to anticipate the most plausible faults in a given program.

Mutation Testing: A mutant is a slightly modified version of the program under test, differing from it by a small syntactic change.

2.5.5 Usage-Based Techniques

Operational Profile: In testing for reliability evaluation (also called operational testing), the test environment reproduces the operational environment of the software, or the operational profile, as closely as possible. The goal is to infer from the observed test results the future reliability of the software when in actual use.

Usability Inspection Methods: Usability principles can provide guidelines for discovering problems in the design of the user interface. Are also called usability inspection methods, including: Heuristic evaluation or User Observation Heuristics, Heuristic estimation, Cognitive walkthrough, Pluralistic walkthrough, Feature inspection, Consistency inspection, Standards inspection and Formal usability inspection.

2.5.6 Model-Based Testing Techniques

Finite-State Machines: By modeling a program as a finite state machine, tests can be selected in order to cover the states and transitions.

Workflow Models: Workflow models specify a sequence of activities performed by humans and/or software applications, usually represented through graphical notations.

2.5.7 Reviews

Walkthrough: The purpose of a systematic walk-through is to evaluate a software product. A walkthrough may be conducted for educating an audience regarding a software product.

Peer Review: The authors do not explain the artifact. They give it to one or more colleagues who read it and give feedback. The aim is to find defects and get comments on the style, including: Team reviews, Walkthroughs, Pair programming, Peer desk check passaround and Ad hoc review.

Technical Review: Further formalize the review process. They are often also management reviews or project status reviews with the aim to make decisions about the project progress. In general, a group discusses the artefacts and decides about the content.

Inspection: The purpose of an inspection is to detect and identify software product anomalies. Some important differentiators of inspections as compared to other types of technical reviews are the roles (author, inspection leader, inspector, and scribe) and the inspection process which consists of the steps planning, kick off, individual checking, logging meeting and edit and follow-up. Some examples of inspections are: Checklist-based reading, Usage-based reading, Defect-based reading, and Perspective-based reading.

2.6 Conclusions

In this chapter, a background on software quality and V&V methods is provided, in which important terms and definitions related to the quality of software are presented, as well as the relationship that exists between them. The evolution of the quality standards of the ISO is analyzed, emphasizing the characteristics of the last one, the ISO 25010 standard, also described in the chapter. Finally, 19 validation and verification methods are classified and characterized, which together with the 8 quality characteristics of ISO 25010, constitute the main objects of studies in the present work. This theoretical framework also served as supporting documentation for the survey form shown in Appendix A.

Chapter 3

Systematic Mapping on Combinations of V&V Methods

3.1 Introduction

In the present chapter it is analyzed the combination approach by “Compilation” as defined Elberzhager [27] in the previous systematic mapping. The authors here [27] describe two fundamental approaches to combining dynamic and static V&V methods: the “compilation” approach based on the selection of a set of V&V techniques, and the approach for “Integration” which is the union of several techniques to form a new one, included papers related.

The study focuses on the analysis of the Compilation approach since the purpose of the study is to treat the combination as the use/selection of a set of techniques of V&V (Compilation of V&V) and not as the combination of different techniques for compose a new one or use the results of the application of an first technique as instances to apply another second (Integration of V&V).

The mapping executed will be a partial update of the first [27]. The snowballing technique is used to obtain new studies from 2010 to the present related to the approach by Compilation. The documents found in the compilation approach of the first SM will be added to the result of the search string, followed by the application of snowballing forwards and backwards.

The chapter presents the detailed planning of the SM protocol, defining the research questions, the search strategy and the selection strategy to be applied to obtain the most relevant bibliography. Followed by the results and their analysis.

3.2 Systematic Mapping

A systematic mapping (SM) is a type of secondary study where the main goal is the classification and thematic analysis of literature on a specific topic. A SM follows the same principled process as systematic literature reviews, but have different criteria for inclusions/exclusions and quality. Due to its broader scope and varying type of studies, the collected data and the synthesis tend to be more qualitative than for systematic literature reviews. However, it is important for the contribution and relevance of a mapping study that the analysis goes beyond the pure descriptive statistics and relates the trends and observations to real-world needs [17].

For this research is fundamental to find, know and classify the bibliography that exists about the combination of methods and techniques of V&V by means of SM to Show the frequency/number of publications per category within a scheme. In this way, the coverage of a particular area of research or topic can be determined.

For the conduction of the SM on the combination of methods and techniques of V&V a protocol was carried out where a series of stages were defined, executed and explained in detail in the following sections.

3.3 Protocol Planning

3.3.1 Research Questions

Given the need to identify, catalog and classify existing literature regarding the combination of V&V methods, the following research questions to organize and summarize the content are formulated:

Main Question RQ1:

What approaches have been proposed for combining verification and validation methods and which it is most suited to the purposes of work?

The objective of this main question is to obtain from the bibliography all the relevant information about the approaches and strategies that exist to combine verification and validation methods and identify which best suits the purposes of work: treating the combination as the use/selection of a set of the V&V techniques.

To support this main question, the following secondary questions were derived:

Secondary Question RQ1.1:

What is the purpose of combining verification and validation methods?

To understand why the need to combine methods in each study. The goals.

Secondary question RQ1.2:

What types of V&V are used to be combined by the approach?

Intended to extract from the selected bibliography the type of V&V methods that are combined by the approach and which methods are identified if specified.

Secondary question RQ1.3:

How are the V&V methods being combined by the approach?

The information to obtain from this question is how the methods presented in each of the studies are combined.

Main Question RQ2:

What are the types of research in which the approaches are presented?

This question responds were the types of research presented in the approaches. In this way, the bibliography can be classified according to Wieringa [77]:

- **Evaluation research:** Implemented in practice, evaluation of implementation conducted; requires more than just one demonstrating case study.
- **Solution proposal:** Solution for a problem is proposed, benefits/application is demonstrated by example, experiments, or student labs; also, includes proposals complemented by one demonstrating case study for which no long-term evaluation/dissemination plan is obvious.
- **Philosophical paper:** New way of thinking, structuring a field in form of a taxonomy or a framework, secondary studies like SLR or SMS.
- **Opinion paper:** Personal opinion not grounded in related work and research methodology.
- **Experience paper:** Personal experience, how are things done in practice.

3.3.2 Search Strategy

3.3.2.1 The PICO Criteria and Keywords

To determine the keywords of the search string and the research questions, is used the **PICO** criteria: (population, intervention, comparison, outcomes) with the purpose of delimiting the scope of the investigation.

According to Kitchenham [50]: The **population** in which the evidence is collected, i.e. which group of people, programs or businesses are of interest for the review? The **intervention** applied in the empirical study, i.e. which technology, tool or procedure is under study? The **comparison** to which the intervention is compared, i.e. how is the control treatment defined? The **outcomes** of the experiment should not only be statistically significant, but also be significant from a practical point of view.

For this study, that give a general view on the combination of V&V methods, in the area of software engineering, the following criteria of the **(PI)CO** are sufficient for de SM:

Population: Research papers about V&V methods.

Keywords: software, approach, method, strategy, technique,

Intervention: Combination of V&V methods.

Keywords: combining, combination, unifying, unification, selecting, selection, verification, validation, inspection, test, testing.

3.3.2.2 The Controls Papers

- M. B. Dwyer and S. Elbaum, **Unifying Verification and Validation Techniques: Relating Behavior and Properties through Partial Evidence**. FSE/SDP workshop on Future of Software Engineering Research (FOSE), pp. 93-98, Santa Fe, New Mexico, USA, 2010.
- D. Cotroneo, R. Pietrantuono and S. Russo, **A Learning-Based Method for Combining Testing Techniques**. Proceedings - International Conference on Software Engineering 6606560, pp. 142-151, San Francisco, CA; USA, 2013.
- S. Mouchawrab, L.C. Briand and Y. Labiche, **Assessing, comparing, and combining statechart-based testing and structural testing: An experiment**. Proceedings - 1st International Symposium on Empirical Software Engineering and Measurement, ESEM, 4343731, pp. 41-50, Madrid; Spain, 2007.

3.3.2.3 Data Search and Snowballing Strategy

Taking as a starting point, the keywords based on the papers of control and defined by (PI)CO criteria, is constructed the search string presented in the 3.1, to support and extend the initial seed set extracted from the first SM [27]. The search should return in the result the controls papers.

On the set of seeds resulting from the search, the snowballing technique was applied backwards (from the reference lists) and forward snowballing (finding quotes from the news papers), with the aim of increasing the scope of the study and find relevant and recent bibliography. The backwards snowballing is repeated in the documents obtained from each iteration, so as not to lose any potentially relevant study, until the new set of paper obtained is empty. It is a recursive process.

To carry out this process, efficient and organized, are followed the instructions of Felizardo et al. [32]. The authors say that it is beneficial to have several search approaches that support the updating of this type of study (SM), such as database searches and snowballing techniques (backward and forward). About forward snowballing, for example, considerably reduces the effort of updating secondary studies in Software Engineering due to its high precision, however, the risk of losing relevant documents should not be underestimated [32].

In the case of the use of search strings to update secondary studies, it is important that the protocol be carefully detailed and documented, including the search string, to perform the replication. Unfortunately, for this update, it is not possible to perform a replication of the first SM [27] with the precision that it should have, since in the protocol specification the search string used is not clear, although it is still a study very well argued that it predicts and validates this type of threats with respect to replications. However, the works resulting from this first SM [27] are fundamental as a starting point for the update.

For all the above, it was decided to make a new protocol and a search chain in accordance with the proposed objectives. In the presented protocol, the key words and their possible combinations are redefined in the most appropriate way, which is a great challenge in this type of studies due to the lack of formalization of the terminology. There is no common terminology, appropriate descriptors and keywords in the area of Software Engineering. This constitutes one of the greatest difficulties faced by specialists, since the probability of using the same term to refer to the same concept is usually very small [32].

Table 3.1: Search String

TITLE-ABS (software AND (approach OR method OR strategy OR technique) AND (((combining OR combination OR unifying OR unification) AND (verification AND validation)) OR ((combining W/3 Testing) OR (combining W/3 Review) OR (combination W/3 Testing) OR (combination W/3 Review) OR (unifying W/3 Testing) OR (unifying W/3 Review) OR (unification W/3 Testing) OR (unification W/3 Review))))

The search string shown in Table 3.1 above was built based on the benefits of the SCOPUS library, using the resources that its advanced search engine offers.

3.3.2.4 Data Sources

To execute the search string, the SCOPUS library is chosen, this is one of the most recommended data sources of papers for software engineering researchers, they provide access to important journals and conferences in the area such as ACM, IEEE, Springer, Elsevier, and it has less foibles than others [16].

For the development of this SM, the benefits and resources offered by the SCOPUS library are exploited to the maximum. In the construction of the string, first, is used “TITLE-ABS”, which means that the documents returned, include in the title and summary the keywords used in the search string. The following resource is also used: “W/3”, allowing between one keyword and another to exist up to three words in the middle.

3.3.3 Selection Strategy

For the selection of the studies it is necessary to establish criteria and make filters since the search return many papers that are not relevant for the research. The inclusion and exclusion criteria and the 5 filters are presented below:

3.3.3.1 Inclusion Criteria

- The language must be English.
- Papers published in peer reviewed venues (e.g., conferences, workshop, and journals)
- Studies related to project development and software engineering.

- Studies focused on the combination of methods of V&V attending the combination strategy by Compilation specifically.
- Parts of the string and the keywords should be contained in the titles and abstracts of the papers.

3.3.3.2 Exclusion Criteria

- White paper, thesis, technical report, power point, proceedings abstracts and irrelevant source.
- Studies in duplicity.
- Studies not being available.
- Studies completely outside the area and research topic.

3.3.3.3 Procedures of Selecting Studies by Filters

The identification and filtering of the papers is divided into five steps following for the followed by the snowballing process, described below. The complete selection process is summarized and represented by the Figure

- Searching for papers using the Search String in the digital library selected for the study. This first step returned 370 papers.
- The first filtering (Filter 1) Use the exclusion criteria in this process. With this filter, is reduced to 174 papers.
- The second filter (Filter 2) Filtering by reading the titles, abstracts and using the inclusion and exclusion criteria. Here is reduced to 32 relevant papers.
- The last filter (Filter 3) Read the selected papers in full. With this last filter, is selected 5 new papers after applying the inclusion and exclusion criteria. In this step, is added only 23 papers corresponding to the “Compilation” approach extracted from the initial systematic mapping [27] of 26 initial papers, because two were not possible to obtain and one not meet the inclusion criteria. A total of 28 paper were obtained show in the Table 3.2 , that use the prefix S for the IDS of the papers obtained by the search string and M for the papers IDS of the first SM [27]. Highlighting the years after 2010, the year to where the documents were analyzed by Elberzhager in [27].

- Finally, apply the Forward and Backward Snowballing, checking the references list (backward snowballing) and finding citations to the papers (forward snowballing), of both group: the papers resulting from the search string and the ones that is extracted from the first Systematic Mapping [27], obtaining 20 new papers, adding up in total 48 papers.

In the Table 3.3 shows the process data of backward (B) with the references and forward (F) with citations snowballing in the first iteration, taking as an initial set of papers the shown above in the Table 3.2. For each paper that corresponds an ID, it is done backward snowballing from the reference lists: REFs (B) and forward snowballing finding citations to the papers: REFs (F). Filters 2: ABSTRACT and 3: FULL READ are applied for the references and citations (B/F) using the inclusion and exclusion criteria. Finally, the new paper of each one is obtained and then, the process is repeated again in another iteration and so on until the set of papers NEWs is empty.

It is important to know that to get the new papers is applying the inclusion and exclusion criteria. In the case of duplicates, as is to be expected, there is always the possibility that more than one study refers to the same paper, in this case, the new paper corresponds to the first one that references it according to the order in which they are analyzed, and so for each iteration.

In the Table 3.4 is showed the results of the first iteration of the snowballing proses, identifying the new papers use the prefix N for their IDs, the TITLE and YEARS of the studies. The column FROM represent the origin of the papers, for example, F if it was for forward snowballing or B if it was for backward snowballing followed by the id of the paper from which they came. This procedure will be performed for each of the subsequent iterations: the second iteration represented in the Table 3.5 and Table 3.6, and the third and final iteration by the Table 3.7. The process concludes when no new results are found.

Table 3.2: Set of Initial Papers

ID	TITLE	YEAR	REF
S1	Unifying Verification and Validation Techniques: Relating Behavior and Properties through Partial Evidence	2010	[26]
S2	A Learning-Based Method for Combining Testing Techniques	2013	[22]
S3	Assessing, comparing, and combining statechart-based testing and structural testing: An experiment	2007	[59]
S4	On the integration of software testing and formal analysis	2012	[14]
S5	Combining Testing and Proof to Gain High Assurance in Software: A Case Study	2013	[9]
M1	A verification-centric software development process for Java	2009	[80]
M2	Combining static and dynamic analysis of concurrent programs	1994	[4]
M3	Enforcing object protocols by combining static and runtime analysis	2008	[36]
M4	HAVE: detecting atomicity violations via integrated dynamic and static analysis	2009	[20]
M5	HEAT: an integrated static and dynamic approach for thread escape analysis	2009	[19]
M6	Integrating static and dynamic analysis for detecting vulnerabilities	2006	[1]
M7	The use and limitations of static-analysis tools to improve software quality	2008	[2]
M8	An empirical evaluation of defect detection techniques	1997	[64]
M9	An empirical evaluation of six methods to detect faults in software	2002	[71]
M10	An empirical evaluation of three defect-detection techniques	1995	[48]
M11	An experimental evaluation of inspection and testing for detection of design faults	2003	[3]
M12	An industrial case study of the verification and validation activities	2003	[8]
M13	Comparing the effectiveness of software testing strategies	1987	[7]
M14	Detection or isolation of defects? An experimental comparison of unit testing and code inspection	2003	[66]
M15	Estimating the value of inspections and early testing for software projects	1994	[51]
M16	Functional testing, structural testing, and code reading: what fault type do they each detect?	2003	[43]
M17	Studying the effects of code inspection and structural testing on software quality	1998	[52]
M18	Test inspected unit or inspect unit-tested code?	2007	[38]
M19	“Continuous verification” in mission critical software development	1997	[18]
M20	A method combining review and testing for verifying software systems	2008	[21]
M21	Comparing bug finding tools with reviews and test	2005	[75]
M22	Integration of formal specification, review, and testing for software component quality assurance	2009	[57]
M23	V&V of flight and mission-critical software	1989	[24]

Table 3.3: First Iteration

ID	REFs (B)	CITEs (F)	ABSTRACT (B/F)	FULL READ (B/F)	NEWS (B/F)
S1	20	4	9/3	3/2	0/0
S2	25	6	8/3	5/2	0/0
S3	25	8	9/4	3/2	0/1
S4	18	2	7/1	3/1	0/0
S5	24	4	6/1	2/0	0/0
M1	41	5	5/2	2/2	0/1
M2	42	6	3/1	0/1	0/0
M3	30	7	4/4	1/2	0/2
M4	21	23	8/11	3/3	0/0
M5	15	2	2/0	0/0	0/0

Table 3.4: Results of First Iteration

ID	TITLE	YEAR	FROM	REF
N1	Model-based quality assurance of automotive software	2008	F-S3	[45]
N2	Combining tests and proofs	2008	F-M3	[35]
N3	Verifying consistency of web services behavior	2008	F-M3	[79]
N4	An integrated analysis and testing methodology to support model-based quality assurance	2014	F-M1	[28]
N5	A Controlled Experiment in Program Testing and Code Walk-throughs/Inspections	1978	B-M8	[60]
N6	A controlled experiment to explore potentially undetectable defects for testing techniques	2014	F-M8	[70]
N7	Determining the effectiveness of three software evaluation techniques through informal aggregation	2013	F-M9	[63]
N8	A comparative analysis of three replicated experiments comparing inspection and unit testing	2011	F-M9	[67]
N9	Modelling the effects of combining diverse software fault detection techniques	2000	F-M9	[55]
N10	What do we know about defect detection methods?	2006	F-M11	[65]
N11	Comparing and combining software defect detection techniques: A replicated empirical study	1997	B-M12	[78]
N12	A replicated empirical study to evaluate software testing methods	2017	F-M13	[73]
N13	Comparing the effectiveness of equivalence partitioning, branch testing and code reading by stepwise abstraction applied by subjects	2012	F-M13	[44]
N14	Are test cases needed? Replicated comparison between exploratory and test-case-based software testing	2014	F-M13	[42]
N15	Further investigations into the development and evaluation of reading techniques for object-oriented code inspection	2002	F-M14	[25]
N16	Integrating specification-based review and testing for detecting errors in programs	2007	B-M20	[56]
N17	Comparing the effectiveness of penetration testing and static code analysis on the detection of SQL injection vulnerabilities in web services	2009	F-M21	[5]
N18	Introduction of static quality analysis in small- and medium-sized software enterprises: experiences from technology transfer	2016	F-M21	[34]

Table 3.5: Second Iteration

ID	REFs (B)	CITEs (F)	ABSTRACT (B/F)	FULL READ (B/F)	NEWS (B/F)
N1	18	1	4/1	3/0	2/0
N2	4	0	2/0	1/0	0/0
N3	21	5	2/0	1/0	0/0
N4	65	0	6/0	2/0	0/0
N5	11	129	3/18	0/6	0/0
N6	9	0	5/0	2/0	0/0
N7	36	1	11/0	5/0	0/0
N8	20	1	3/0	1/0	0/0
N9	16	1	4/0	2/0	0/0
N10	22	59	8/12	2/4	0/0
N11	15	36	5/10	0/3	0/0
N12	28	0	6/0	0/0	0/0
N13	58	17	9/3	2/0	0/0
N14	70	9	10/2	3/1	0/0
N15	19	18	0/3	0/0	0/0
N16	17	15	2/2	0/1	0/0
N17	21	30	1/2	0/0	0/0
N18	74	7	4/2	0/0	0/0
Total	524	329	85/55	24/15	2/0

Table 3.6: Results of Second Iteration

ID	TITLE	YEAR	FROM	REF
N19	A comparison of three verification techniques: Directed testing, pseudo-random testing and property checking	2002	B-N2	[6]
N20	Experimental evaluation of verification and validation tools on martian rover software	2004	B-N2	[15]

Table 3.7: Third Iteration

ID	REFs (B)	CITEs (F)	ABSTRACT (B/F)	FULL READ (B/F)	NEWS (B/F)
N19	3	29	0/7	0/2	0/0
N20	19	38	3/9	1/3	0/0
Total	22	67	3/16	1/5	0/0

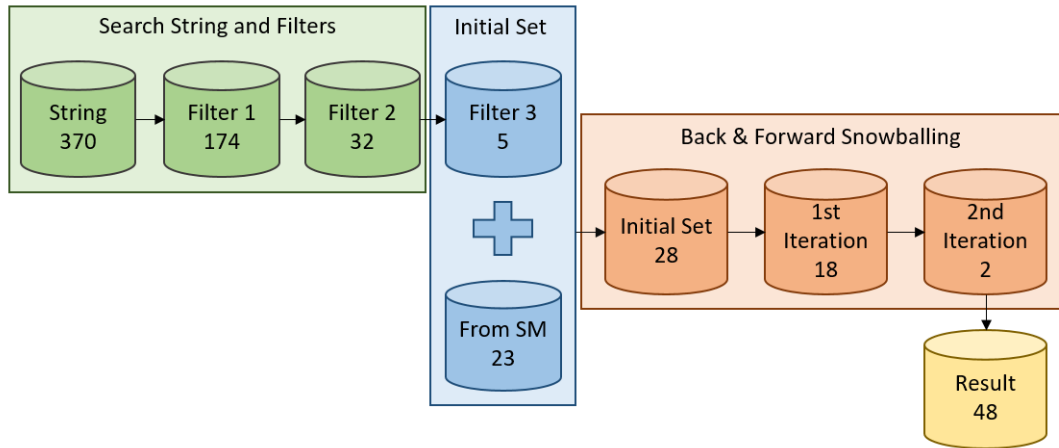


Figure 3.1: Studies selection process

3.4 Results

3.4.1 Data Extraction Strategy

At this stage, is collected the information from the 48 selected papers, answering the research questions.

RQ1: *What approaches have been proposed for combining verification and validation methods and which it is most suited to the purposes of work?*

Two categories of approaches are defined by Elberzhager [27] to combine V&V methods and their corresponding subcategories:

Compilation approach: treat the combination as the process of selecting V&V methods. This is divided into three subcategories. The first, static and dynamic analysis combined, the second compares different testing techniques and inspections discussing advantages and disadvantages among them and the third describes other combinations for example some techniques of testing each other, inspections combined with formal specifications, bug-finding tools among others. Important to note that the 48 paper obtained in the SM correspond to this category.

Integration approach: address the combination as the process of creating a new technique combining different methods in one or using the results of the application of some technique as an instance to apply another one technique in consequence. This is divided into two subcategories: the first is the integration of static and dynamic analysis which covers most of the approaches. With respect to the order of application, the

static analysis with greater frequency is first to be applied, followed by dynamic analysis. The second subcategories, integrate inspection and testing approaches, where inspections derive test cases and the test code that is automatically generated is inspected.

RQ1.1: *What is the purpose of combining verification and validation methods?* With the purpose of **evaluate the effectiveness and efficiency** are the papers: S2, S3, M8, M10, M11, M13, M16, M17, M18, N3, N9, N11, N12, N13, N14 and N17. To **detect the most amount of problem and faults**: M3, M4, M5, M6, M14, N7, N8, N16, N19 and N20. To **complement the results**: S1, M2, M9, M18, N1, N2, N5, N10, N15 and N18, this means that the results of applying a method is complemented by the results of the others that are combined, since what cannot be detected by one, the other can detect it. **To reduce efforts, costs and runtime**: S5, M7, M15, M21 and M22. With other specific purposes such as **provide a roadmap for future research** the study: S4. To **verify and analyze a software development process and ensure quality**: M1 and N4. To **investigate the balance between inspections and tests** M12. To **ensure software quality** from the early stages of development M19. To **maintain the consistency** between the specification structure and the program structure M20. To **guarantee the security, qualification and validity of the system**: M23 and with the purpose of determine which types of defect are potentially undetectable to either one: N6.

RQ1.2: *What types of V&V are used to be combined by the approach?*

To answer this research question, new subcategories were redefined within the compilation approach regarding to the type of V&V methods. Each subcategory corresponds to a group of types of methods that are usually combined. The Figure 3.2 represents in percentage the corresponding peppers in each group, they are:

- **Static & Dynamic Analysis:** S1, S5, M1, M2, M3, M4, M5, M6, M7, N2, N4, N15, N17, N18 and N20.
- **Reviews–Inspection & Testing:** M9, M11, M12, M13, M14, M15, M16, M17, M18, M20, M21, N5, N8, N10 and N16.
- **Testing & Testing:** S2, S3, S4, M8, M10, N1, N6, N7, N11, N12, N13, N14 and N19.
- **Formal Methods & Others:** M19, M22, M23, N3 and N9.

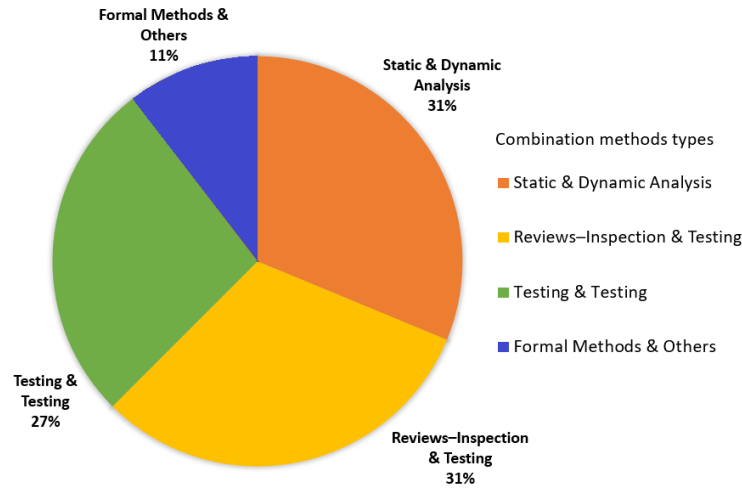


Figure 3.2: Combinations methods type

RQ1.3: *How are the V&V methods being combined by the approach?*

The approaches are combined and classified in three categories fundamentally, by **Tools:** S1, S4, M1, M2, M3, M4, M5, M6, M7, and N17, such as framework and algorithms. By **Comparison:** S2, S3, M8, M9, M10, M11, M12, M13, M14, M15, M16, M17, M18, M21, M22, N1, N2, N5, N6, N7, N8, N10, N11, N12, N13, N14, N15, N16, N18, N19 and N20, which consists in the analysis of the results of each of the methods used, complementing in most cases. And finally in the classification of **Others:** S5, M19, M20, M23, N3, N4 and N9, those who have a very particular way of combining for each of the studies. The Figure 3.3 represents in percentage the corresponding peppers in each combination category.

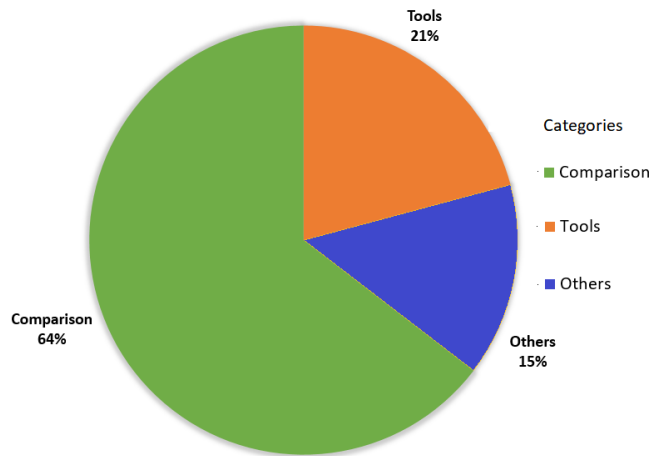


Figure 3.3: Combinations categories

RQ2: *What are the types of research in which the approaches are presented?*

Three types of research, including the empirical studies that determine it, were identified in the papers.

- **Evaluation Research** determined by empirical study “Case Study” includes the papers: S1, S2, S5, M1, M12, M14, M19, M21, M22, M23, N2, N3, N18, N19.
- **Philosophical Paper** determined by empirical study “Secondary Study” include the paper: S3.
- **Solution Proposal** determined by the empirical studies “Example” including the papers: M2, M3, M7, M20, N1, N4, N5, N9, N16, N17. And the empirical studies “Experiment” including the papers: S3, M4, M5, M6, M8, M9, M10, M11, M13, M14, M16, M17, M18, N6, N7, N8, N10, N11, N12, N13, N14, N15, N20.

The Figure 3.4 shows the percentages that represent the types of research and the empirical studies of the total of papers, sticking out the **Solution Proposal** as the type of research most used together with the empirical study “Experiment”, taking the greater amount of the papers analyzed.

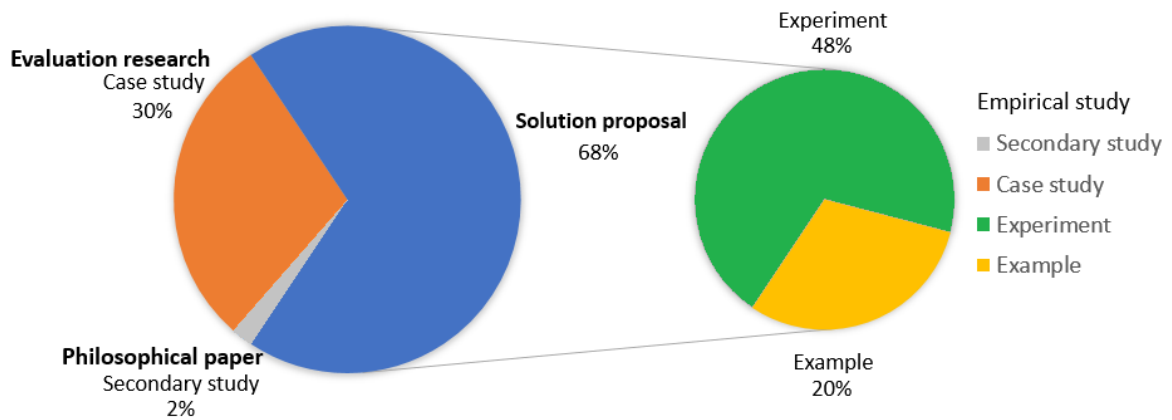


Figure 3.4: Research type

3.4.2 Data Synthesis Strategy

In this stage, an analysis of the data was done through graphs to compare effect sizes and values to assess the synthesized outcome.

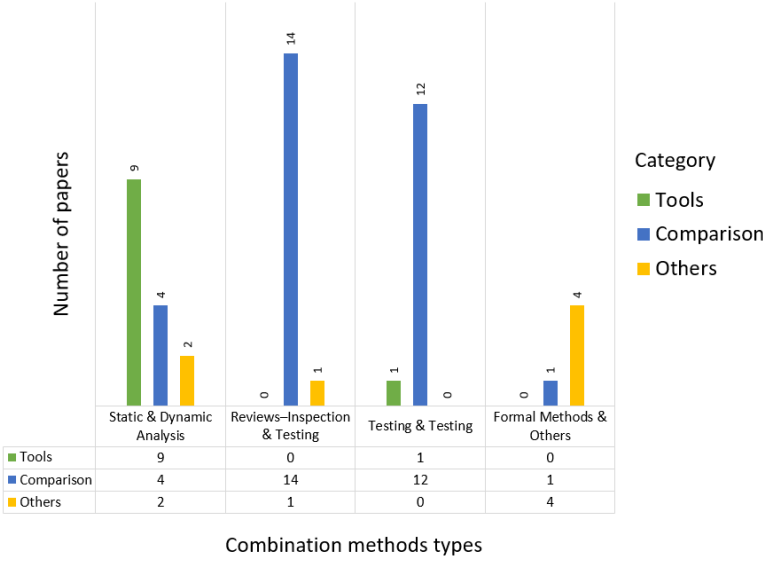


Figure 3.5: Number of papers by combination methods types

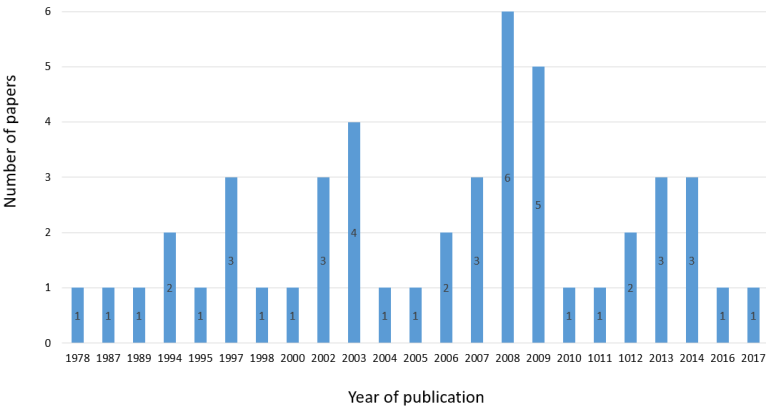


Figure 3.6: Number of papers by years of publication

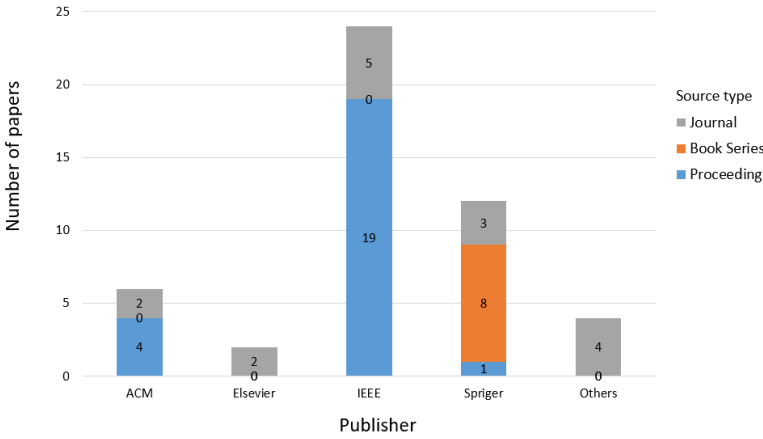


Figure 3.7: Number of papers by publisher and source type

Figure 3.5 represents the number of paper by type of combination of methods and category of combination, previously defined in the research questions. The graph shows the highest concentration of studios in combinations of the type Reviews-Inspection & Testing (14 paper) through the comparison, followed by Testing & Testing (12 paper) in the same category.

Figure 3.6 shows the number of papers per year of publication, reflecting a crescent development of research on the years between 2000-2009.

Finally the Figure 3.7, number of papers by publisher and source type, indicates that the IEEE is the publisher with the more number of studies published in conference proceeding compared to the others, with an amount of 24 paper representing the half of the total papers analyzed in this SM.

3.5 Conclusions

In this chapter, a SM of V&V methods was performed. 48 relevant papers with approaches that combine V&V techniques were found. In most of them, the type of V&V methods and the purpose of the combination were identified. These papers were classified by type of empirical study, type of research and approach they used. For all this is can conclude that the SM fulfilled the proposed goal.

The snowballing proved to be, once again, effective and the most appropriate search technique for updates of systematic studies. This SM, in addition, can be considered a potential guide for future studies of this type in the academic field, being able to apply its protocol, meticulously detailed and explained, to any context.

Chapter 4

Relating V&V Methods to Software Product Quality Characteristics: Results of an Expert Survey

4.1 Introduction

The main goal of the survey presented is to obtain concrete evidence on which V&V methods are the most appropriate to cover each of the ISO 25010 characteristics, from the point of view of the 19 software engineering experts, all of them with a PhDs in Software Engineering and Quality, being part of outstanding committees in the area and having relevant publications. The results gives an initial configuration of V&V methods. The initial configuration indicates which of the previously selected methods can be used to address each one of the software quality characteristics of the ISO 25010 standard. It can serve as a further reference in the software development industry.

The planning of the survey is explained in detail in each of its stages according to Wohlin [17]: Main goal and Scope of the study using the GQM (goal, question, metric) definition template. The population, research questions, metrics, the execution strategy, the statistical techniques for data analysis, the instrumentations and the validity assessment are also described in the planning. In the chapter the operation is described also, survey operation, i.e., how the survey was conducted, as well as survey results are presented and analyzed.

4.2 Survey Planning

A Survey is a process of collection of information from people to describe, compare or explain their knowledge, attitudes and behavior. Its main tools to collect qualitative or quantitative data are interviews or questionnaires made to people from a representative sample of a population. It is realized to obtain information relevant for some investigation [17].

4.2.1 Main Goal and Scope

The main goal of this survey is to gather initial evidence, through expert opinion, about the suitability of V&V methods to address ISO 25010 software product quality characteristics. Using the GQM (Goal Question Metric) definition template [17] this goal can be stated as: **Analyze V&V methods for the purpose of characterization with respect to their suitability for addressing ISO 25010 software quality characteristics from the point of view of experts in the area of V&V in the context of the software engineering research community.**

4.2.2 Population

Following the advice of deciding upon the target population based on whether they are the most appropriate to provide accurate answers instead of focusing on hopes to get high response rates [72], the population of V&V experts will be sampled by selecting PhDs in software engineering that are active in at least one of the following software engineering and V&V program committees: ICSE, ICST, ESEM, SEAA-SPPI, and SWQD. Additionally, each survey participant should have at least one publication within the last 5 years directly related to V&V methods. While there are other sampling strategies, this one can be allows effectively reaching a sample of V&V experts from the software engineering research community

4.2.3 Survey Questions

The survey was designed with only two very direct questions. The intent of keeping the design simple was to allow the experts to answer within a reasonable timeframe.

Q1: To what extent do you agree that the following V&V methods can be applied to address the listed quality attributes?

This question was structured as a table crossing the selected V&V methods (cf. Section 2) against the ISO 25010 quality characteristics (cf. Section 3). The researchers should provide their answers filling each cell with a number corresponding to a Likert scale (1- Disagree, 2-Partially Disagree, 3- Partially Agree, 4- Agree, and N- Not Sure).

Q2: Complete the list by rating any other V&V methods that you believe could be applied to address one or more of the listed quality attributes.

This question was optional and provided to allow the expert to suggest and evaluate other V&V methods, not included in the initial list, that he considers relevant.

4.2.4 Metrics

Likert scales (1- Disagree, 2-Partially Disagree, 3- Partially Agree, 4- Agree, and N- Not Sure) were used for assessing the V&V methods against the quality characteristics in both questions. The aggregated metric on the agreement for each method/quality-characteristic set was obtained using the median value, which can be safely applied to **Likert scales** [17].

4.2.5 Execution Strategy

The execution strategy consisted of identifying the population sample according to the strategy and distributing the survey instrument via email. Due to the format of the questions, the survey was provided by e-mail as an MS Word attachment. Participants should answer the survey within 15 days.

4.2.6 Statistical Techniques

The aggregation of the responses for the set of answers was conducted using the median value. Additionally, the Median Absolute Deviation (MAD), representing the degree of concordance between the experts, should be analyzed to further understand the representativeness of the median for the sample. Statistical visualization features to provide an overview of the results include tables and a bubble plot crossing information of V&V methods and quality characteristics.

4.2.7 Instrumentation

The questionnaire instrument had a title, a short description of the research goal, a note of consent stating that individual data will be handled anonymously, followed by the two questions. Additionally, as supporting documentation, short descriptions of the V&V methods and the ISO 25010 quality characteristics were also provided as an appendix.

4.2.8 Validity Assessment

Throughout the process of planning the survey, is identified some threats. Table 4.1 lists these potential threats and how are treated them in the survey. One of the main mitigation strategies was validating the instrument by asking other individuals to answer the survey, as part of a pilot study, before handing it over to the experts. This also allow to understand that, to keep it is simple as possible, answering the questionnaire still requires at least 20 minutes.

Table 4.1: Threats and Treatment

Threats	Treatment
Bad instrumentation.	Revision and evaluation of the questionnaire about the format and formulation of the questions. Running a pilot study.
Inadequate explanation of the constructions.	Revision and evaluation of the questionnaire about the format and formulation of the questions. Running a pilot study.
Doubts of the experts on the purpose or on specific definitions.	Including the research goal explanation and adding support information on the V&V methods and ISO 25010 quality characteristics..
Measurement and results reliability.	Using medians to aggregate individual Likert scale entries. Using the median absolute deviation to check on the agreement among the experts..
Statistical conclusion validity.	This threat strongly depends on the sample size. A mitigation that could be used is running future survey replications and aggregating the results.

4.3 Operation

The population sampling strategy, described in the sub-section 4.2.2, allowed to identify 145 candidate subjects (PhDs in software engineering, active in one of the selected program committees, and with relevant publications on V&V methods). The survey was sent to them by e-mail as an MS Word attachment, which could be easily answered by using any MS Word compatible editor. Experts had 15 days to answer the survey. After this deadline the data collection was considered concluded.

At all, 19 experts (response rate of 13%) from 7 different countries answered the survey. Taking into account the main factors that directly affect the response rate [54]: length of the form (number of pages), sending of the form through e-mail, duration to fill out the form (indeed, some experts mentioned that answering took them much longer than expected from our pilot study) and comparing with similar studies (e.g., [30]), where the response rate is commonly around 10%, it can be considered that the response rate satisfactory and according to the expectation.

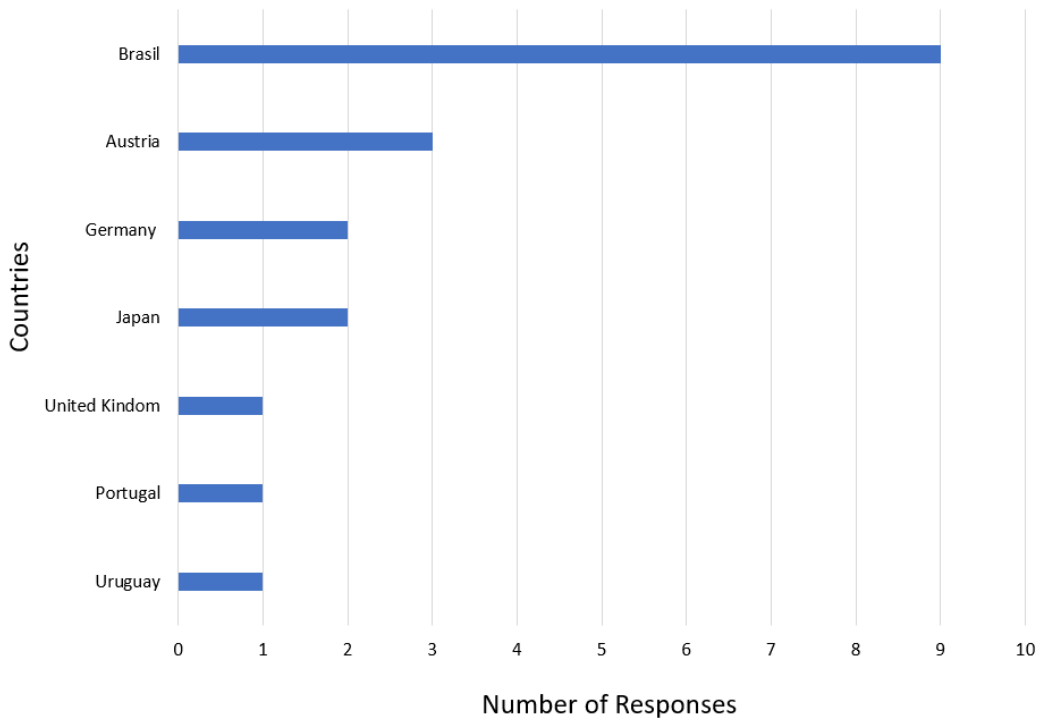


Figure 4.1: Number of responses by country

The Figure 4.1 represents the number of responses per country. It can be observed that most of the answers came from Brazil and Austria. This was probably related to the direct relationship of the authors with researchers from these countries.

4.4 Survey Results

The overall results relating the V&V and ISO 25010 quality characteristics are shown in Table 4.2. The rows contain each of the selected V&V methods and the columns show the ISO 25010 software quality characteristics: FS–Functional Suitability, PE–Performance Efficiency, C–Compatibility, U–Usability, R–Reliability, S–Security, M–Maintainability and P–Portability. The numbers in the cells correspond to the median values of the expert’s answers on how suitable the method is for a given characteristic. For the purpose of calculating the median value ‘N - Not Sure’ responses were not considered. The Median Absolute Deviations (MAD) are shown within parentheses.

For the analysis of the survey data, is considered that the method is reported to address a quality characteristic if the median value of the answers of the respondents is greater than or equal to 3. The cells corresponding to these values are highlighted in Table 4.2. It is possible to observe that, according to the experts, there is at least one of the selected V&V methods addressing each quality characteristic. Most of the methods address functional suitability.

The MAD represents the agreement between the experts. Figure 4.2 shows the overall distribution of the MAD values: 32%, 9%, 58%, and 1%, for the MAD values 0 (blue), 0.5 (orange), 1 (grey), and 1.5 (yellow), respectively. It can be seen that these values mainly oscillate in a range between 0 and 1 (except for one element that equals 1.5, concerning using inspections to address reliability). These overall low deviations indicate small differences between the opinions of the experts.

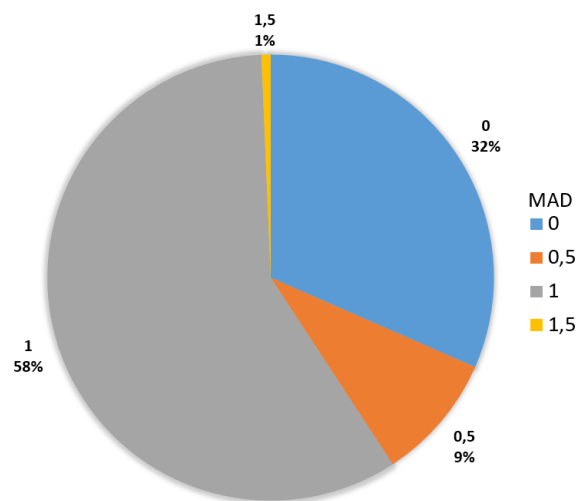


Figure 4.2: Degree of concordance (MAD) between the experts

Table 4.2: Survey Result

Methods	FS	PE	C	U	R	S	M	P
1.Ad Hoc	3 (0)	1 (0)	2 (0.5)	3 (1)	1 (0)	1.5 (0.5)	1 (0)	1 (0)
2.Exploratory Testing	3 (1)	2 (1)	2 (0.5)	3 (1)	2 (0)	2 (1)	2 (0)	2 (1)
3.Equivalence Partitioning	4 (0)	2 (1)	1.5 (0.5)	1 (0)	2 (1)	1 (0)	1 (0)	1 (0)
4.Pair wise Testing	3.5 (0.5)	1 (0)	2 (1)	1 (0)	2 (1)	2 (1)	1 (0)	1 (0)
5.Boundary-Value Analysis	4 (0)	2 (1)	2 (1)	1 (0)	2 (1)	2 (1)	1 (0)	1 (0)
6.Random Testing	3 (1)	2 (1)	2 (1)	1 (0)	2 (1)	2 (1)	1 (0)	1 (0)
7.Cause-Effect Graphing	4 (0)	2 (1)	2 (0.5)	1.5 (0.5)	2 (1)	2 (1)	1.5 (0.5)	1 (0)
8.Control Flow-Based	3 (1)	1.5 (0.5)	1 (0)	1 (0)	3 (1)	2 (1)	2 (1)	1.5 (0.5)
9.Data Flow-Based	3 (1)	1 (0)	1 (0)	1 (0)	3 (1)	2 (1)	2 (1)	1 (0)
10.Error Guessing	3 (1)	2 (1)	2.5 (0.5)	2 (1)	3 (1)	2 (1)	1 (0)	2 (1)
11.Mutation Testing	3 (1)	1 (0)	2 (1)	1 (0)	3 (1)	2 (1)	1 (0)	1 (0)
12.Operational Profile	3 (1)	3 (1)	3 (1)	3 (1)	3 (1)	2 (1)	1 (0)	2 (1)
13.Usability Inspection	2 (1)	1.5 (0.5)	2 (1)	4 (0)	2 (1)	2 (1)	1 (0)	1 (0)
14.Finite-State Machines	4 (0)	2 (1)	2 (1)	2 (1)	3 (1)	3 (1)	2 (1)	1 (0)
15.Workflow Models	4 (0)	2 (0)	2 (1)	3 (1)	2 (1)	3 (1)	2 (1)	1 (0)
16.Walkthrough	3 (1)	2 (1)	2 (0)	2 (1)	2 (1)	2 (1)	3 (0.5)	2 (1)
17.Peer Review	3 (1)	2 (1)	2 (1)	2 (1)	2 (1)	3 (1)	3 (1)	2 (1)
18.Technical Review	3 (0)	2 (1)	2 (1)	2 (1)	2.5 (0.5)	3 (1)	3 (1)	3 (1)
19.Inspection	4 (0)	2 (1)	2 (1)	3 (1)	2.5 (1.5)	3 (1)	3 (1)	3 (1)

4.4.1 Answering the Research Question 1

RQ1: *To what extent you agree that the following V&V methods can be applied to address the listed quality attributes?*

The figure 4.3 provides a summary of the relation between the V&V methods and the quality characteristics in a bubble plot. In this Figure, the size of the bubble: small, medium and large, represents the median value: 3, 3.5 and 4, respectively. The colors refer to the MAD value: Blue, Orange and Grey represent the values of 0, 0.5 and 1, respectively. Thus, the large blue plots represent combinations where the experts agree (median 4) with strong consensus (MAD 0) that the V&V method is suitable for addressing the quality characteristic. It is noteworthy that the grey dots, still represent a positive evaluation for the combination (median 3 and MAD 1).

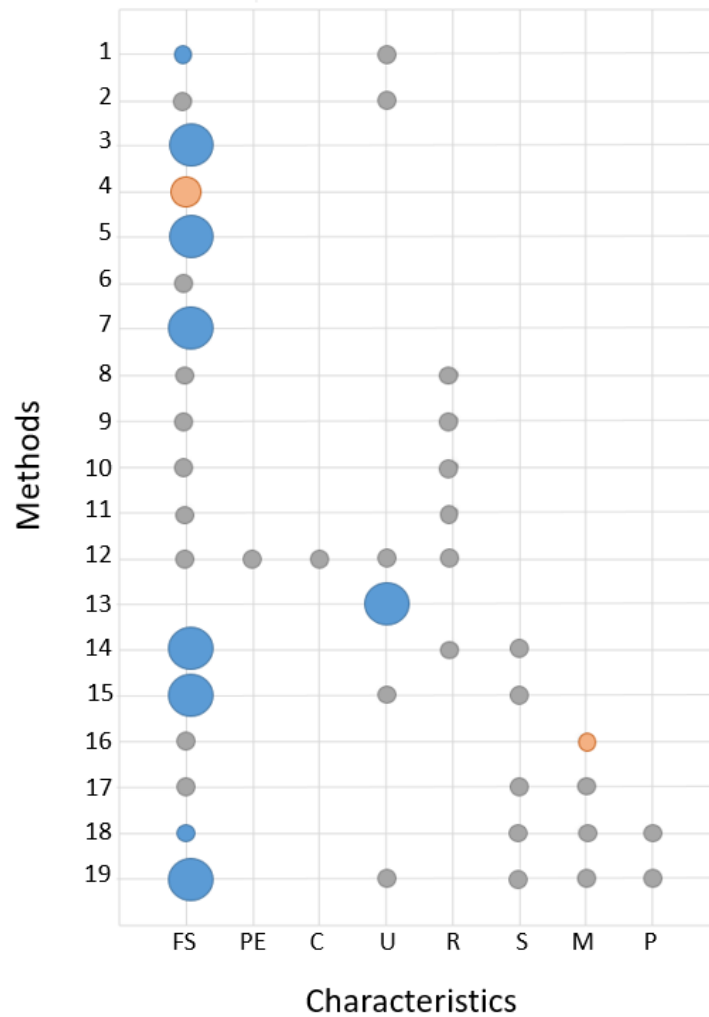


Figure 4.3: Results of the first question

4.4.2 Answering the Research Question 2

RQ2: *Complete the list by rating any other V&V methods that you believe could be applied to address one or more of the listed quality attributes.*

The participants mentioned 20 other (more specific) V&V methods. Among these methods, the ones that were cited by more than one expert were: **Model Checking**, **Penetration Testing**, **Stress Testing**, and **Fuzz Testing**. It is noteworthy that all methods suggested by more than one expert are automated or semi-automated ones. This indicates that in a future survey trial such methods should probably be included.

4.5 Conclusions

This chapter proposed to establish a relation between a set of V&V methods and the ISO 25010 quality characteristics, based on expert opinions. It is noteworthy that, to the best of knowledge, such relation, while being extremely relevant for research and practice, has not yet been established.

Therefore, an initial list of V&V methods has been compiled and carefully selected experts to answer the survey. At all, It is received answers from 19 experts, all holding PhDs in software engineering, being part of relevant program committees and having recent publications concerning V&V methods.

The resulting relations (suitability of the V&V methods to address the ISO 25010 quality attributes) are summarized in Figure 4.3. All the bubbles represented in this Figure concern an agreement on the relation between the method and the quality characteristic. Nevertheless, considering the sample size, the relations depicted by the small (median 3) grey (MAD 1) bubbles, while still representing an aggregated expert agreement on the relationship, should be taken with a grain of salt, given that in these cases the experts disagreed slightly more. Considering the overall sample, experts mostly provided consistent answers with small deviations from the median.

While the sample size is small and do not claim for statistical conclusion validity and are aware of the importance of replications to reinforce the results, is still believe that the aggregated opinion of 19 experts can serve as a starting point for other researchers and practitioners, who currently completely lack information on the suitability of V&V methods to address ISO 25010 quality attributes.

Chapter 5

An Efficient Algorithm for Combining V&V Methods

5.1 Introduction

In this Chapter, an algorithm to obtain the optimal combination of methods in reasonable computational time is presented. With the purpose of find the optimal solution for the problem, is adopted a parameterized approach, considering the set of quality characteristics as a parameter, and obtaining an algorithm classified as FPT (fixed-parameter tractable) in the Parameterized Complexity Theory. This field emerged as a promising alternative for working with NP-hard problems [68].

The algorithm runs in $O(f(k) \times n)$, where the constant k is the number of quality characteristics, n is the number of methods, and $f(k)$ is some function of k . Considering that the number of quality characteristics of a given quality standard is always constant, the algorithm runs in polynomial time in terms of the number of V&V methods to be combined.

It is also presented, a brief introduction to the theory of complexity parameterized, the problem is modeled as an SCP, followed by the FPT-algorithm, the experiment are execute and analyzed. Finally, a discussion about the approach is made.

5.2 Parameterized Complexity

The Parameterized Complexity field emerged as a promising approach way to deal with NP-hard problems [68]. It is a branch of the Computational Complexity Theory that focuses on classifying computational problems according to their hardness with respect

to different parameters of the input. The complexity of a problem is mainly expressed through a function of these parameters.

The theory of NP-completeness was developed to identify problems that cannot be solved in polynomial time whether $P \neq NP$. However, it is fundamental to identify problems for which a polynomial time algorithms for their resolution is not expected, several NP-complete and NP-hard problems still need to be solved in practice.

For many problems, only super-polynomial time algorithms are known when the complexity is measured according to the size of the input, and in general, they are considered “intractable” from the theoretical point of view assuming that P is different from NP . Nevertheless, for several problems is can develop algorithms in which is can split its their running time into a part computed in polynomial time with respect to the size of the input and another part computed in exponential time or worse, but only with respect to a parameter k isolated to analysis. Consequently, if is set the parameter k to a small value and its growth is relatively small could be consider these problems as “manageable” and not “intractable” [23] [33] [62].

Thus, an important question arises: “Do these hard problems admit non-polynomial time algorithms whose exponential complexity part is a function of merely some aspects of the problem?” [68]. The existence of such algorithms was analyzed by Downey and Fellows in [23], and is briefly discussed in the next section.

5.2.1 Fixed-Parameter Tractable (FPT) Approach

The fixed-parameter tractable (FPT) approach [23] considers the following format for of the problems: “Given an object x and a non-negative integer k , the goal is to determine whether x has some property that depends on k ?” The parameter k is considered small compared to the size of x . The relevance of these parameters lies precisely in the small range of values they can take, being a very important factor in practice [68].

The FPT-algorithms sacrifice the execution time, which can be exponential, but guarantees that the exponential dependency is restricted to the parameter k , which means that the problem can be solved efficiently for small values of that fixed parameter. The use of these algorithms provides a more rigorous analysis of problem’s time complexity since this complexity is generally obtained from the size of the input [68].

Formally, a problem Π belongs to the class FPT (it is fixed-parameter tractable) with respect to a parameter k if it admits an algorithm to solve it whose running time is of the

form: $f(k) \times n^a$, where a is a constant, and f is an arbitrary computable function. Note that whenever k is bounded by a constant it has $O(f(k) \times n)$, hence the running time of the algorithm will be polynomial.

Finally, for the problem in question, is presented a fixed-parameter tractable algorithm where the size k of the set of characteristics to be covered is the parameter. I.e., the complexity is being limited by the number of relevant product characteristics to be considered when developing the software.

5.2.2 Scalability of the FPT-algorithms

Scalability is the ability of a system or process to handle an increasing amount of data [11]. Computer algorithms can be called scalable if they are efficient when applied to large instances, i.e., instances with a large size of the input [53].

It can say that FPT-algorithms are scalable because they are efficient when executed in large instances. These algorithms take advantage of the specific structure of the instances, which is a differential when comparing to exact or exhaustive search algorithms that require high computational time.

It is important to note that the studied problem can handle a large number of V&V methods, given that the number of quality characteristics tends to be relatively small. Therefore, an FPT-algorithm with respect to the number of characteristics to be covered will produce a tool for combination of V&V methods with high scalability.

Indeed, in the problem, the number of quality characteristics is already a known small integer (in the ISO standard this number is 8). Therefore, scalability relies on the ability of finding the optimal solution even if the number of considered methods is growing. The initial set comprises 19 methods, but additional methods have been reported by the survey respondents and the algorithm allows to efficiently work, with 50, 100, 200, 500 and even with 1000 methods as shown by the results of the experiments in Table X for these instances.

5.3 Modeling the Problem

The problem of finding the smallest combination of methods that cover a specific set of quality characteristics can be modelled as a Set Cover Problem.

Considering C as the set of characteristics, and M_i as the subset of C that is covered

by a specific method. Is needed to find the smallest set of subsets that cover C . The problem is NP-hard in general. The relation between the characteristics and the methods can be modeled as an undirected bipartite graph. The Figure 5.1 shows an instance of such graph. In the left-hand side the methods are positioned, and in the right-hand side the characteristics. Edges reflect the relationship between methods and characteristics.

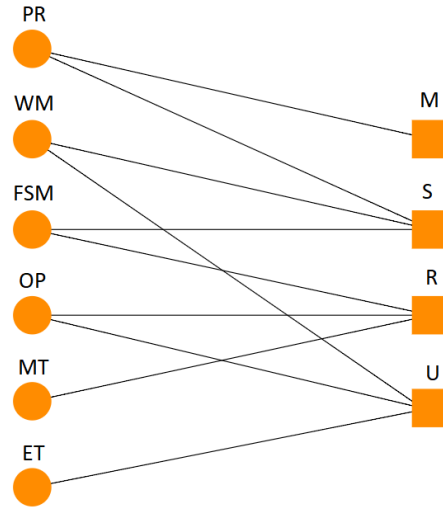


Figure 5.1: Bipartite graph of V&V methods vs. Quality characteristics

In the depicted instance, the set of methods contains the following elements: Peer Review (PR), Workflow Models (WM), Finite-State Machines (FSM), Operational Profile (OP), Mutation Testing (MT), and Exploratory Testing (ET). The set of characteristics is composed by four characteristics: Usability (U); Reliability (R); Security (S), and Maintainability (M). The graph shows a scenario in which the method PR covers characteristics M and S, WM covers S and U, FSM covers S and R, OP covers R and U, MT covers R, and ET covers U.

The example instance was obtained from the results of a survey presented in the Chapter 4 – Table 4.2 that gathered the opinion of experts on these V&V methods, for this only a subset of the data will be taken to explain with more clarity and accuracy the execution of the algorithm. In the survey, the experts answered about their agreement on the suitability of the methods to address the different quality attributes of the ISO 25010 standard. The relationship between some method m and some characteristic c is obtained from the median of the survey answers (1 – disagree, 2 – partially disagree, 3 – partially agree, 4 – agree). In this example is considered that m properly covers c if the median is bigger or equal to 3, for example, only methods that cover a quality characteristic to a certain degree will have edges in the graph. Of course, this threshold for the initial configuration of the graph can be adjusted, but edges should not be included for methods

that badly cover a certain quality characteristic (or only some minor aspect of it).

The example serves for illustrative purposes to present the V&V method combination algorithm. Actually, this shorter example allows providing a better understanding of the algorithm's execution and correctness.

5.4 FPT–Algorithm to Combine V&V Methods

The goal of the algorithm is to obtain the optimal combination (smallest number) of V&V methods that properly cover all the relevant quality characteristics for the product to be developed. Certainly, a software organization could complement the resulting set with other V&V methods that cover similar quality characteristics to find more defects and to further enhance quality, but at least they would know the minimum set of methods to consider in order to address all the quality characteristics that are relevant for the product to be developed. I.e., a combination such that there is a method properly addressing (i.e., with an edge in the graph for) each relevant quality characteristic and none of them remains uncovered.

The objective function is the number of selected methods that properly cover all the characteristics. The parameter to be set is the number of the selected quality characteristics. In this way, the Set Cover Problem is being parameterizing by the number of characteristics to be covered by the V&V methods. Coming up next, is presented some definitions that are used in the algorithm:

C – set of characteristics.

M – set of methods.

$N(m)$ – set of characteristics covered by the method m .

$P(c) = \{x \in M : c \in N(x)\}$ – set of methods that cover the characteristic c .

$R(m) = \{x \in M : N(x) \subseteq N(m)\}$ – set of methods that cover a subset of $N(m)$.

Figure 5.2 shows the pseudocode of the FPT–Algorithm. The input parameters are the set of characteristics C and the set of methods M . In lines 1 and 2, the variables M^* and f^* are initialized with the smallest set of methods found until now (initially, all methods), and the optimal value of the objective function found until now (number of methods in M). If the set of characteristics C is empty, then an empty set of methods is returned in lines 3-4.

Algorithm 1 Set Cover Algorithm, **Parameters:** sets C, M

```

1:  $M^* \leftarrow M$ 
2:  $f^* = |M|$ 
3: if  $C = \emptyset$  then
4:   return  $\emptyset$ 
5: else
6:    $M \leftarrow \text{RemoveSubsets}(M)$ 
7:    $c \leftarrow \text{SelectCharacteristic}(C)$ 
8:    $C_t \leftarrow C \setminus \{c\}$ 
9:   for all  $m \in P(c)$  do
10:     $M' \leftarrow \{m\}$ 
11:     $C_t \leftarrow C_t \setminus N(m)$ 
12:     $M_t \leftarrow M \setminus R(m)$ 
13:    while  $\exists c' \in C_t : |P(c') \cap M_t| = 1$  do
14:      Let  $m' \in P(c')$ 
15:       $C_t \leftarrow C_t \setminus N(m')$ 
16:       $M_t \leftarrow M_t \setminus \{m'\}$ 
17:       $M' \leftarrow M' \cup \{m'\}$ 
18:    end while
19:     $M'^* \leftarrow \text{SetCover}(C_t, M_t)$ 
20:    if  $|M'^*| + |M'| < f^*$  then
21:       $f^* \leftarrow |M'^*| + |M'|$ 
22:       $M^* \leftarrow M'^* \cup M'$ 
23:    end if
24:     $C_t \leftarrow C \setminus \{c\}$ 
25:  end for
26: end if
27: return  $M^*$ 

```

Figure 5.2: Set cover FPT-Algorithm

Otherwise, the redundant methods are removed in line 6 by using a simple preprocessing step. It removes methods that cover a subset of characteristics covered by any other method. A characteristic c is selected from the set of characteristics in line 7. The algorithm then focuses on selecting the method that will cover c in the optimal solution. In line 8, the variable C_t that contains the characteristics to be covered is initialized. A loop runs through all the methods that cover c in lines 9-25. The set M' that stores the methods that will be part of a feasible solution is initialized with method m in line 10. The set of characteristics to cover C_t is updated in line 11 by removing the characteristics already covered by m . The set M_t , containing the methods available to cover C_t , is initialized in line 12 with all methods of M except those covering a subset of $N(m)$. In

lines 13-18 a loop is executed while there are characteristics c' that are covered by a single method m' . The variables C_t , M_t and M' are updated in lines 15-17. The available M_t methods and the characteristics that have not been covered until now are used to obtain an optimal sub-problem solution by recursively calling the SetCover algorithm. In line 19, the obtained optimal solution is stored in M'^* . If the methods selected in M' together with the optimal solution M'^* of the sub-problem improve the optimum value found so far f^* , then f^* and M^* are updated in lines 20-23. The value of C_t is reinitialized in line 24. The best solution found (M^*), is returned as the optimal solution to the problem in line 27.

5.4.1 Execution of the Set Cover Algorithm

Taking the graph represented in Figure 5.1 as the entry of the algorithm, we now illustrate the execution of the pseudocode. After initialization steps 1-5, line 6 removes redundant methods. In this case, methods MT and ET are removed, because they cover only one characteristic, already covered by other methods. The result is shown in Figure 5.3.

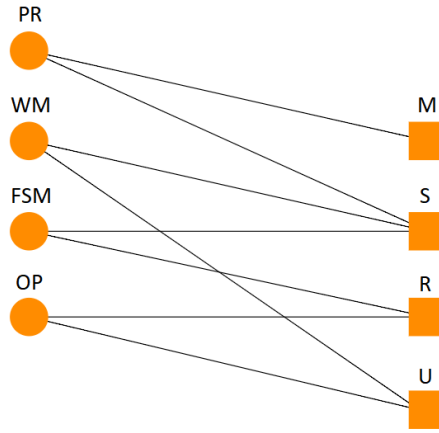


Figure 5.3: Algorithm execution. State of the graph after the preprocessing step

Afterwards, the first characteristic M is chosen as c , and all the methods that cover M must be considered in the cycle that begins in line 9. Therefore, method PR is selected. In line 11, we remove all the characteristics already covered by PR, that is, M and S. The variable M_t gets the set of methods WM, FSM, OP in line 12. Since there are no characteristics covered by only one method, loop in lines 13-18 does not perform any action, and the algorithm is called recursively in line 19 with set of characteristics R, U, and set of methods WM, FSM, OP as parameters. Figure 5.4 illustrates the graph at this stage.

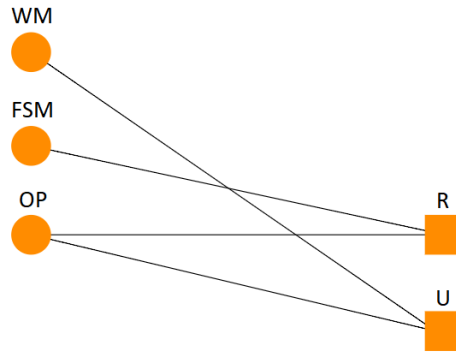


Figure 5.4: Algorithm execution. State of the graph after the recursive call

Finally, the algorithm is executed again from the beginning. Methods WM and FSM are immediately removed as redundant, and the remaining method OP is selected to cover the last two characteristics. The variables C_t and M_t became empty, and in the next recursive call, stopping criterion is reached. The OP method is returned as a solution of the instance represented in Figure 5.4, forming the final solution of the whole instance together with already selected method PR. The smallest set of methods M^* is set as PR, OP, and the optimal value f^* is set to 2.

Figure 5.5 shows the methods that form the optimal solution returned by the algorithm when executed in the graph. If M is selected as the first characteristic at the beginning of the execution, then the optimal set of methods returned by the algorithm is PR, OP.

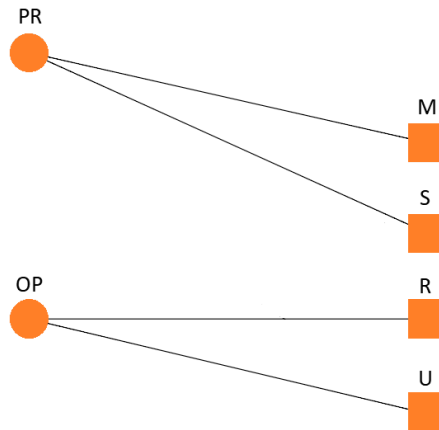


Figure 5.5: Optimal solution

As can be observed from the execution, the algorithm considers all the possible ways of covering the quality characteristics, keeping the most efficient ones. In this sense, the obtained solution can be considered as optimal for the problem and model we pose.

In fact, the algorithm is able to determine the optimal combination (smallest number) of V&V methods that properly cover all the relevant quality characteristics for a product to be developed based on any initial graph configuration connecting V&V methods to the quality characteristics they properly address.

5.4.2 Running Time Analysis

Suppose that there are n methods in the set M , and there are k characteristics in the set C , being k some small integer. Can be note that a naive (brute force) algorithm would test all solutions (subsets of the set M) and chose which of them cover C having smaller size. Because there are 2^n subsets of the set M , this naive algorithm has a time complexity of $O(2^n)$. This exponential order is intractable even for some relatively small values of n , like 30 or 40.

Instead, the proposed algorithm tries to determinate which method is the best option to cover each characteristic. After choosing some characteristic c , the algorithm tries to select each method that properly covers c , covering the rest of characteristics recursively. Because each characteristic can be covered by any of the n methods, the order of this algorithm can be initially bounded by $O(n^k)$. In general, this order is already better than the 'naive' solution.

Nevertheless, is improved the upper bound of our algorithm's running time by refining the actual number of methods it will analyze. In fact, there are only 2^k different ways of covering a set C of k elements. If there is more than 2^k methods, then necessarily there will be two of that cover exactly the same set of characteristics. That means that these two methods would be similar and indistinguishable to our algorithm; that is, if they cover the same characteristics, you can use any of them at your chose. Using this fact, is can successively preprocess the input, improving the algorithm performance from $O(n^k)$ to $O(f(k) \times n)$, where $f(k) \leq (2^k)^k = 2^{k^2}$. In our case, $k = 8$ and this means that $f(k)$ is upper bounded by a constant, i.e., $f(k) = O(1)$. Then, a linear algorithm for the problem instead of an exponential or even a $O(n^8)$ -time algorithm is have.

Once again, the upper bound for $f(k)$ is improved (decreased) using the fact that if some method m_1 covers a subset of characteristics covered by some other method m_2 , then m_1 can be removed from the set of methods. This is because if m_1 is actually chosen, you can instead choose m_2 , since m_2 is "better" method in the sense that it covers all that m_1 covers, and possibly more. Lubell [58] showed that there no more than $\binom{k}{\lfloor k/2 \rfloor}$ combinations with the property that no one is a subset of the other. This implies that

for $k = 8$, there can be much less than 2^k different methods with the property that there is no method that covers a subset of characteristic of some other method. In particular, for $k = 8$ there can be at most $\binom{8}{4} = 70$ methods satisfying this property, and at most $\binom{7}{3} = 35$ of these methods attending a common characteristic. Therefore, the redundant methods are removed by using a simple preprocessing step that searches for methods that cover a subset of characteristics covered by any other method. At each iteration of the algorithm, the number of characteristics to be covered decreases and the previous steps of the algorithm are repeated considering a decremented k value.

Summarizing, it holds that:

$$f(k) < \binom{k-1}{\lfloor (k-1)/2 \rfloor} \times \binom{k-2}{\lfloor (k-2)/2 \rfloor} \times \cdots \times \binom{2}{1} \quad (5.1)$$

For $k = 8$, it follows that $f(k) < 35 \times 20 \times 10 \times 6 \times 3 \times 2$, then our $O(f(k) \times n)$ -time algorithm is an efficient (linear) algorithm where $f(k)$ is upper bounded by a constant. In practice, this constant is even lower, and does not depend on the number of existing methods, which produces scalability with respect to the number of methods to be worked.

5.5 Computational Experiments

Several experiments were performed to assess the validity of the algorithm presented above. The algorithm was implemented in C# programming language and compiled by Roslyn, a reference C# compiler, in an Intel Core i3 machine with a 2.0 GHz processor and 4 GB of random-access memory, running under the Windows 10 operating system.

A number of test problems created by a random generator is considered. Each test problem has two parameters: the number of vertices n and the probability p of a method to cover a characteristic.

The FPT-algorithm is also executed on the instance obtained from the survey described in Chapter 4, and detailed in Table 4.2.

5.5.1 Integer Linear Programming Model

The following classical Integer Linear Programming (ILP) model for the Set Cover problem was implemented in CPLEX program.

$$\begin{aligned}
& \min \sum_{j \in M} x_j \\
& \text{subject to: } \sum_{j \in M} a_{ij} x_j \geq 1, \quad \forall i \in C. \\
& x_i \in \{0, 1\}, i \in C,
\end{aligned}$$

where a_{ij} indicates if the characteristic i is covered by the method j , and the variable $x_j = 1$ if the method j is selected, $x_j = 0$ otherwise.

This ILP model is used to verify the correctness of the FPT-Algorithm, as well as to validate the performance of the method.

5.5.2 Results

Table 5.1 shows the optimal solution sizes and runtimes for FPT-Algorithm solver and CPLEX, for each instance. The name of the instance indicate the number of methods, followed by the probability of a characteristic to be covered by a method, in percents. The optimal solution sizes (number of methods returned) are equal for all instances, indicating the correctness of the FPT-Algorithm.

Both, the CPLEX solver for the ILP model, and the FPT-Algorithm are efficient, obtaining the result in less than 0.01 seconds in all cases. The CPLEX solver is mostly faster, but the small computational times even for big instances make the difference in time negligible for applications.

For the instance obtained from the survey described in Chapter 4, and detailed in Table 4.2, the FPT-Algorithm returned the methods: 12, 19 and the CPLEX returned the methods: 12, 18. The solution returned by the FPT-Algorithm contains the method 19 that covers a superset of the characteristics covered by the method 18 in the CPLEX solution, showing that FPT-Algorithm performs better using this additional comparison criterion.

Table 5.1: Experiment Results

Instance	Runtime FPT-Algorithm	Optimal Solution FPT-Algorithm	Runtime CPlex	Optimal Solution CPlex
Instance_20_2	0.00031	No solution	0.00013	No solution
Instance_20_5	0.00016	No solution	0.00013	No solution
Instance_20_10	0.00031	5	0.00013	5
Instance_20_20	0.00047	3	0.00011	3
Instance_20_50	0.00062	2	0.00011	2
Instance_50_1	0.00016	No solution	0.00011	No solution
Instance_50_2	0.00016	No solution	0.00011	No solution
Instance_50_5	0.00016	5	0.00011	5
Instance_50_10	0.00031	5	0.00013	5
Instance_50_20	0.00110	2	0.00014	2
Instance_50_50	0.00094	2	0.00013	2
Instance_100_1	0.00094	No solution	0.00009	No solution
Instance_100_2	0.00047	No solution	0.00013	No solution
Instance_100_5	0.00062	5	0.00014	5
Instance_100_10	0.00094	2	0.00016	2
Instance_100_20	0.00125	3	0.00013	3
Instance_100_50	0.00110	2	0.00019	2
Instance_200_1	0.00063	7	0.00014	7
Instance_200_2	0.00078	8	0.00019	8
Instance_200_5	0.00078	4	0.00013	4
Instance_200_10	0.00344	3	0.00013	3
Instance_200_20	0.00188	2	0.00014	2
Instance_200_50	0.00094	1	0.00014	1
Instance_500_1	0.00141	8	0.00016	8
Instance_500_2	0.00156	5	0.00016	5
Instance_500_5	0.00312	4	0.00014	4
Instance_500_10	0.00359	3	0.00014	3
Instance_500_20	0.00469	2	0.00014	2
Instance_500_50	0.00234	1	0.00016	1
Instance_1000_1	0.00344	7	0.00016	7
Instance_1000_2	0.00281	4	0.00017	4
Instance_1000_5	0.00531	3	0.00016	3
Instance_1000_10	0.00766	3	0.00017	3
Instance_1000_20	0.00578	2	0.00016	2
Instance_1000_50	0.00500	1	0.00014	1

5.6 Discussion

For illustrative purposes, the described example was based on the initial results of an expert survey (analyzed in Chapter 4). This initial configuration is out of the scope of applications in industry, since companies could use a configuration based on a set of V&V methods that they use, or on their own expert beliefs. This initial configuration may be considered as a reference and starting point to be refined to the convenience of companies or users.

From a practical point of view, companies might decide to complement the optimal solution provided by the algorithm by applying additional V&V methods that cover similar quality characteristics (e.g., aiming at finding additional defects and further enhancing product quality), in particular for critical projects. However, using this approach at least they would know the minimum set of methods to consider in order to properly address the quality characteristics that are relevant for the product to be developed.

On the other hand, the specialists on software engineering economics might argue that the solution providing the smallest number of V&V methods is not considering the cost of applying each method. However, to address this issue is needed to know the relative cost among the V&V methods and this information is extremely context specific and hard to generalize. This is considered a limitation that could be addressed more thoroughly in future work. A possible solution to handle this problem when using the described approach could be the elimination of methods that are cost-limiting in the initial configuration.

5.7 Conclusions

The proposed FPT-Algorithm promises to be effective, being able to provide the optimal combination (smallest number) of V&V methods properly covering a set of chosen quality characteristics to be considered when developing a software product. Additionally, it is more efficient than brute-force or exhaustive search algorithms and its execution time properties match the particularities of the problem well. Indeed, the algorithm can be applied to instances of different sizes, making the approach scalable, i.e., suitable for larger case studies (for instance, considering more V&V methods).

It is noteworthy also that none of the related work cited in the Chapters 1 and 2 has implemented something similar to this proposal, since its focus is on covering all set of quality

characteristics with few methods, thus obtaining a general and optimal combination of V&V methods. While applying all available methods represents a solution, this option might not be applicable due to cost constraints.

Chapter 6

Concluding Remarks

6.1 Conclusions

For the study of literature and related works a systematic mapping was performed. 49 relevant papers with approaches that combine V&V methods were found. In most of them, the type of V&V methods to combine were identified. This systematic mapping classified the papers founded by empirical study, research and approach of combinations they used. The main search technique applied, snowballing, proved to be effective and the most appropriate for updates of secondary studies.

Whit the survey, an initial set of V&V methods that can be used in software development projects to evaluate and guarantee quality with respect to quality characteristics of the ISO 25010 standard was identified. A survey for the characterization of those V&V methods in relation to the quality characteristics was planned and executed using opinions of experts. The results of the survey are essential for the identification of “best” V&V methods that address a set of quality characteristics.

The problem of finding best-quality combination of V&V methods as the SCP, a NP-hard combinatorial optimization problem was modeled. A parameterized FPT–Algorithm that is specially designed for the instances presented was defined, since typically the number of considered characteristics is small. The proposed algorithm is able to provide the optimal combination (smallest number) of V&V methods properly covering a set of chosen quality characteristics to be considered when developing a software product. Additionally, is showed that it is more efficient than brute-force or exhaustive search algorithms in terms of expertise and maintainability. Furthermore, the algorithm can be applied to instances of different sizes, making the approach scalable, i.e., suitable for

larger case studies (for instance, considering more V&V methods).

For all the above mentioned, it can be concluded that the present work fulfilled its main goal and the specific objectives, developing an efficient approach for the optimal combination of V&V methods.

6.2 Contributions

- The answers to the survey allow to obtain the first concrete evidence about the relationship between methods and quality characteristics. The initial configuration that organizations can take as a starting point to be adapted to its context and the type of project.
- A FPT-Algorithm was implemented, the first of its type to solve the SCP using the principles of parameterized complexity, being scalable, maintainable and expert than the others brute-force or exhaustive search algorithms.
- The systematic mapping could be considered a guide for future secondary studies of this type in the academic, since it provides a protocol, meticulously detailed and explained step by step, which can be applied to any context. Highlights the value of the snowballing technique to reduce efforts when update of a secondary study is carrying out.

6.3 Future Work

As future work concerns to consider the use of the application costs of the methods, as well as to extend the initial set of methods with the suggestions proposed by the experts. Also, tool support could be provided to, given to set V&V methods, quality characteristics and an initial configuration (starting from the results of the survey and refining it according to the context and type of project in which the approach will be applied), provide the optimal combination of V&V methods. Apply the approach to different study cases.

References

- [1] AGGARWAL, A.; JALOTE, P. Integrating static and dynamic analysis for detecting vulnerabilities. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)* (2006), vol. 1, pp. 343–350.
- [2] ANDERSON, P. The use and limitations of static-analysis tools to improve software quality. *CrossTalk-Journal of Defense Software Engineering* 21 (06 2008).
- [3] ANDERSSON, C.; THELIN, T.; RUNESON, P.; DZAMASHVILI, N. An experimental evaluation of inspection and testing for detection of design faults. In *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* (2003), pp. 174–184.
- [4] ANGER, F. D.; RODRIGUEZ, R. V.; YOUNG, M. Combining static and dynamic analysis of concurrent programs. In *Proceedings 1994 International Conference on Software Maintenance* (1994), pp. 89–98.
- [5] ANTUNES, N.; VIEIRA, M. Comparing the effectiveness of penetration testing and static code analysis on the detection of sql injection vulnerabilities in web services. In *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing* (2009), pp. 301–306.
- [6] BARTLEY, M. G.; GALPIN, D.; BLACKMORE, T. A comparison of three verification techniques: Directed testing, pseudo-random testing and property checking. In *Proceedings of the 39th Annual Design Automation Conference* (New York, NY, USA, 2002), DAC '02, ACM, pp. 819–823.
- [7] BASILI, V. R.; SELBY, R. W. Comparing the effectiveness of software testing strategies. *IEEE Transactions on Software Engineering SE-13*, 12 (1987), 1278–1296.
- [8] BERLING, T.; THELIN, T. An industrial case study of the verification and validation activities. In *Proceedings. 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No.03EX717)* (2003), pp. 226–238.
- [9] BISHOP, P.; BLOOMFIELD, R.; CYRA, L. Combining testing and proof to gain high assurance in software: A case study. In *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)* (2013), pp. 248–257.
- [10] BOEHM, B.; BASILI, V. R. Software defect reduction top 10 list. *Computer* 34, 1 (Jan. 2001), 135–137.
- [11] BONDI, A. B. Characteristics of scalability and their impact on performance. In *Proceedings of the 2Nd International Workshop on Software and Performance* (New York, NY, USA, 2000), WOSP '00, ACM, pp. 195–203.

- [12] BOURQUE, P.; FAIRLEY, R. E. *SWEBOK: Guide to the Software Engineering Body of Knowledge*, version 3.0 ed. IEEE Computer Society, Alamos, CA, 2014.
- [13] BOZENA POKSINSKA, JENS JORN DAHLGAARD, M. A. The state of iso 9000 certification: a study of swedish organizations. *The TQM Magazine* 14, 5 (2002), 297–306.
- [14] BRAIONE, P.; DENARO, G.; PEZZÈ, M. On the integration of software testing and formal analysis. In *Empirical Software Engineering and Verification: International Summer Schools, LASER 2008-2010, Elba Island, Italy, Revised Tutorial Lectures* (Berlin, Heidelberg, 2012), B. Meyer and M. Nordio, Eds., Springer Berlin Heidelberg, pp. 158–193.
- [15] BRAT, G.; DRUSINSKY, D.; GIANNAKOPOULOU, D.; GOLDBERG, A.; HAVELUND, K.; LOWRY, M.; PASAREANU, C.; VENET, A.; VISSER, W.; WASHINGTON, R. Experimental evaluation of verification and validation tools on martian rover software. *Formal Methods in System Design* 25, 2 (Sep 2004), 167–198.
- [16] BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.* 80, 4 (Apr. 2007), 571–583.
- [17] C. WOHLIN, P. RUNESSON, M. H. M. O. B. R. A. W. *Introduction to Experimentation in Software Engineering*. Kluwer Academic Publishers, Boston, MA, 2000.
- [18] CHANG, T.-F.; DANYLYZSN, A.; NORIMATSU, S.; RIVERA, J.; SHEPARD, D.; LATANZE, A.; TOMAYKO, J. “continuous verification” in mission critical software development. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences* (1997), vol. 5, pp. 273–284 vol.5.
- [19] CHEN, Q.; WANG, L.; YANG, Z. Heat: An integrated static and dynamic approach for thread escape analysis. In *2009 33rd Annual IEEE International Computer Software and Applications Conference* (2009), vol. 1, pp. 142–147.
- [20] CHEN, Q.; WANG, L.; YANG, Z.; STOLLER, S. D. Have: Detecting atomicity violations via integrated dynamic and static analysis. In *Fundamental Approaches to Software Engineering* (Berlin, Heidelberg, 2009), M. Chechik and M. Wirsing, Eds., Springer Berlin Heidelberg, pp. 425–439.
- [21] CHEN, Y.; LIU, S.; WONG, W. E. A method combining review and testing for verifying software systems. In *2008 International Conference on BioMedical Engineering and Informatics* (2008), vol. 2, pp. 827–831.
- [22] COTRONEO, D.; PIETRANTUONO, R.; RUSSO, S. A learning-based method for combining testing techniques. In *2013 35th International Conference on Software Engineering (ICSE)* (2013), pp. 142–151.
- [23] DOWNEY, R. G.; FELLOWS, M. R. *Parameterized complexity, Monographs in Computer Science*. Springer, 1999.
- [24] DUKE, E. L. V&v of flight and mission-critical software. *IEEE Software* 6, 3 (1989), 39–45.

- [25] DUNSMORE, A.; ROPER, M.; WOOD, M. Further investigations into the development and evaluation of reading techniques for object-oriented code inspection. In *Proceedings of the 24th International Conference on Software Engineering* (2002), ICSE '02, ACM, pp. 47–57.
- [26] DWYER, M. B.; ELBAUM, S. Unifying verification and validation techniques: Relating behavior and properties through partial evidence. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (New York, NY, USA, 2010), FoSER '10, ACM, pp. 93–98.
- [27] ELBERZHAGER, F.; MÄNNICH, J.; NHA, V. T. N. A systematic mapping study on the combination of static and dynamic quality assurance techniques. *Information and Software Technology* 54, 1 (2012), 1 – 15.
- [28] ELBERZHAGER, F.; ROSBACH, A.; BAUER, T. An integrated analysis and testing methodology to support model-based quality assurance. In *Software Quality. Model-Based Approaches for Advanced Software and Systems Engineering* (Cham, 2014), Springer International Publishing, pp. 135–154.
- [29] ENDRES, A.; ROMBACH, H. D. *A handbook of software and systems engineering: empirical observations, laws and theories*. Harlow, England ; New York : Pearson/Addison Wesley, 2003.
- [30] FELDERER, M.; AUER, F. Software quality assurance during implementation: Results of a survey in software houses from germany, austria and switzerland. In *Software Quality. Complexity and Challenges of Software Engineering in Emerging Technologies - 9th International Conference, SWQD 2017, Vienna, Austria, January 17-20, 2017, Proceedings* (2017), pp. 87–102.
- [31] FELDT, R.; TORKAR, R.; AHMAD, E.; RAZA, B. Challenges with software verification and validation activities in the space industry. In *2010 Third International Conference on Software Testing, Verification and Validation* (2010), pp. 225–234.
- [32] FELIZARDO, K. R.; MENDES, E.; KALINOWSKI, M.; SOUZA, E. F.; VIJAYKUMAR, N. L. Using forward snowballing to update systematic reviews in software engineering. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (New York, NY, USA, 2016), ESEM '16, ACM, pp. 53:1–53:6.
- [33] FLUM, J.; GROHE, M. *Parameterized complexity theory*. Springer, 2006.
- [34] GLEIRSCHER, M.; GOLUBITSKIY, D.; IRLBECK, M.; WAGNER, S. Introduction of static quality analysis in small and medium-sized software enterprises: Experiences from technology transfer. *CoRR abs/1611.07560* (2016).
- [35] GOPINATHAN, M.; NORI, A.; RAJAMANI, S. Combining tests and proofs. In *Proceedings of the 2Nd International Conference on Verified Software: Theories, Tools, Experiments* (Berlin, Heidelberg, 2008), VSTTE '08, Springer-Verlag, pp. 4–5.

- [36] GOPINATHAN, M.; RAJAMANI, S. K. Enforcing object protocols by combining static and runtime analysis. In *Proceedings of the 23rd ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications* (New York, NY, USA, 2008), OOPSLA '08, ACM, pp. 245–260.
- [37] GOTTERBARN, D.; MILLER, K.; ROGERSON, S.; BARBER, S.; BARNES, P.; BURNSTEIN, I.; DAVIS, M.; EL-KADI, A.; FAIRWEATHER, N.; FULGHUM, M.; JAYARAM, N.; JEWETH, T.; KANKO, M.; KALLMAN, E.; LANGFORD, D.; LITTLE, J.; MECHLER, E.; NORMAN, M.; PHILLIPS, D.; PRINZIVALLI, P.; SULLIVAN, P.; WECKERT, J.; WEIL, V.; WEISBAND, S.; WERTH, L. Software engineering code of ethics and professional practice. *Science and Engineering Ethics* 7, 2 (2001), 231–238.
- [38] GUPTA, A.; JALOTE, P. Test inspected unit or inspect unit tested code? In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)* (2007), pp. 51–60.
- [39] IAN, S. *Software Engineering*, 9th edition ed. Addison Wesley, University of St Andrews, Scotland, 2011.
- [40] ISO. International organization for standardization. Official site: <https://www.iso.org/standards.html>.
- [41] ISO25010. Software product quality, 2011. Official site: <http://iso25000.com/index.php/en/iso-25000-standards/iso-25010>.
- [42] ITKONEN, J.; MÄNTYLÄ, M. V. Are test cases needed? replicated comparison between exploratory and test-case-based software testing. *Empirical Software Engineering* 19, 2 (Apr 2014), 303–342.
- [43] JURISTO, N.; VEGAS, S. Functional testing, structural testing and code reading: What fault type do they each detect? In *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET* (Berlin, Heidelberg, 2003), R. Conradi and A. I. Wang, Eds., Springer Berlin Heidelberg, pp. 208–232.
- [44] JURISTO, N.; VEGAS, S.; SOLARI, M.; ABRAHAO, S.; RAMOS, I. Comparing the effectiveness of equivalence partitioning, branch testing and code reading by stepwise abstraction applied by subjects. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation* (2012), pp. 330–339.
- [45] JÜRJENS, J.; REISS, D.; TRACHTENHERZ, D. Model-based quality assurance of automotive software. In *Model Driven Engineering Languages and Systems* (Berlin, Heidelberg, 2008), Springer Berlin Heidelberg, pp. 858–873.
- [46] KALINOWSKI, M.; MENDES, E.; TRAVASSOS, G. H. An industry ready defect causal analysis approach exploring bayesian networks. In *Software Quality. Model-Based Approaches for Advanced Software and Systems Engineering* (Cham, 2014), D. Winkler, S. Biffl, and J. Bergsman, Eds., Springer International Publishing, pp. 12–33.
- [47] KALINOWSKI, M.; TRAVASSOS, G. H. Ispis: From conception towards industry readiness. In *Chilean Society of Computer Science, 2007. SCCC '07. XXVI International Conference of the* (2007), pp. 132–141.

- [48] KAMSTIES, E.; LOTT, C. M. An empirical evaluation of three defect-detection techniques. In *Software Engineering — ESEC '95* (Berlin, Heidelberg, 1995), W. Schafer and P. Botella, Eds., Springer Berlin Heidelberg, pp. 362–383.
- [49] KARP, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department* (Boston, MA, 1972), R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds., Springer US, pp. 85–103.
- [50] KITCHENHAM, B. A. *Guidelines for performing Systematic Literature Reviews in software engineering (version 2.3)*. Technical Report, EBSE Technical Report EBSE-200701, Keele University and Durham University, 2007.
- [51] L. A. FRANZ, J. C. S. Estimating the value of inspections and early testing for software projects. *Hewlett-Packard Journals* (12 1994).
- [52] LAITENBERGER, O. Studying the effects of code inspection and structural testing on software quality. In *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257)* (1998), pp. 237–246.
- [53] LAUDON, K.; TRAVER, C. *E-commerce: Business, Technology, Society*. Stanford University, 2008.
- [54] LINAKER, J., S. S. M. M. D. M. R. . H. M. Guidelines for conducting surveys in software engineering. *Research Portal of Lund University* (2015).
- [55] LITTLEWOOD, B.; POPOV, P. T.; STRIGINI, L.; SHRYANE, N. Modeling the effects of combining diverse software fault detection techniques. *IEEE Transactions on Software Engineering* 26, 12 (2000), 1157–1167.
- [56] LIU, S. Integrating specification-based review and testing for detecting errors in programs. In *Formal Methods and Software Engineering* (Berlin, Heidelberg, 2007), M. Butler, M. G. Hinchey, and M. M. Larrondo-Petrie, Eds., Springer Berlin Heidelberg, pp. 136–150.
- [57] LIU, S.; TAMAI, T.; NAKAJIMA, S. Integration of formal specification, review, and testing for software component quality assurance. In *Proceedings of the 2009 ACM Symposium on Applied Computing* (New York, NY, USA, 2009), SAC '09, ACM, pp. 415–421.
- [58] LUBELL, D. A short proof of sperner's lemma. In *Classic Papers in Combinatorics* (Boston, MA, 1987), I. Gessel and G.-C. Rota, Eds., Birkhäuser Boston, pp. 402–402.
- [59] MOUCHAWRAB, S.; BRIAND, L. C.; LABICHE, Y. Assessing, comparing, and combining statechart- based testing and structural testing: An experiment. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)* (2007), pp. 41–50.

- [60] MYERS, G. J. A controlled experiment in program testing and code walk-throughs/inspections. *Commun. ACM* 21, 9 (Sept. 1978), 760–768.
- [61] MYERS, G. J.; SANDLER, C.; BADGETT, T. *The art of software testing*. John Wiley & Sons, Hoboken, N.J., 2012.
- [62] NIEDERMEIER, R. *Invitation to λ -parameter algorithms*. Oxford University Press, 2006.
- [63] OLORISADE, B. K.; VEGAS, S.; JURISTO, N. Determining the effectiveness of three software evaluation techniques through informal aggregation. *Information and Software Technology* 55, 9 (2013), 1590 – 1601.
- [64] ROPER, M.; WOOD, M.; MILLER, J. An empirical evaluation of defect detection techniques. *Information and Software Technology* 39, 11 (1997), 763 – 775. Evaluation and Assessment in Software Engineering.
- [65] RUNESON, P.; ANDERSSON, C.; THELIN, T.; ANDREWS, A.; BERLING, T. What do we know about defect detection methods? [software testing]. *IEEE Software* 23, 3 (2006), 82–90.
- [66] RUNESON, P.; ANDREWS, A. Detection or isolation of defects? an experimental comparison of unit testing and code inspection. In *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003*. (2003), pp. 3–13.
- [67] RUNESON, P.; STEFIK, A.; ANDREWS, A.; GRONBLUM, S.; PORRES, I.; SIEBERT, S. A comparative analysis of three replicated experiments comparing inspection and unit testing. In *2011 Second International Workshop on Replication in Empirical Software Engineering Research* (2011), pp. 35–42.
- [68] SANTOS, V.; SOUZA, U. Uma introdução à complexidade parametrizada. *Anais da 34 Jornada de Atualização em Informática, CSBC 2015* (2015), 232–273.
- [69] SHULL, F.; CARVER, J.; TRAVASSOS, G. H. An empirical methodology for introducing software processes. *SIGSOFT Softw. Eng. Notes* 26, 5 (Sept. 2001), 288–296.
- [70] SOLARI, M.; MATALONGA, S. A controlled experiment to explore potentially undetectable defects for testing techniques. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE* (07 2014), vol. 2014.
- [71] SUP, S. S.; DEOK, C. S.; J., S. T.; RAE, K. Y. An empirical evaluation of six methods to detect faults in software. *Software Testing, Verification and Reliability* 12, 3 (2002), 155–171.
- [72] TORCHIANO, M.; FERNÁNDEZ, D. M.; TRAVASSOS, G. H.; DE MELLO, R. M. Lessons learnt in conducting survey research. In *Proceedings of the 5th International Workshop on Conducting Empirical Studies in Industry* (Piscataway, NJ, USA, 2017), CESI '17, IEEE Press, pp. 33–39.
- [73] UMAR, F. S.; S.M.K., Q.; NESAR, A. A replicated empirical study to evaluate software testing methods. *Journal of Software: Evolution and Process* 29, 9 (2017), e1883.

- [74] WAGNER, S. *Software Product Quality Control*. Springer, 2013.
- [75] WAGNER, S.; JURJENS, J.; KOLLER, C.; TRISCHBERGER, P. Comparing bug finding tools with reviews and tests. In *Testing of Communicating Systems* (Berlin, Heidelberg, 2005), F. Khendek and R. Dssouli, Eds., Springer Berlin Heidelberg, pp. 40–55.
- [76] WIEGERS, K. E. *Peer Reviews in Software: A Practical Guide*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002.
- [77] WIERINGA, R.; MAIDEN, N.; MEAD, N.; ROLLAND, C. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.* 11, 1 (2005), 102–107.
- [78] WOOD, M.; ROPER, M.; BROOKS, A.; MILLER, J. Comparing and combining software defect detection techniques: A replicated empirical study. In *Software Engineering — ESEC/FSE’97* (Berlin, Heidelberg, 1997), M. Jazayeri and H. Schauer, Eds., Springer Berlin Heidelberg, pp. 262–277.
- [79] YIN, Y.; LI, Y.; DENG, S.; JIAN, W. Verifying consistency of web services behavior. In *Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, APSCC* (Yilan, Taiwan, 12 2008), IEEE Asia-Pacific Services, pp. 1308–1314.
- [80] ZIMMERMAN, D. M.; KINIRY, J. R. A verification-centric software development process for java. In *2009 Ninth International Conference on Quality Software* (2009), pp. 76–85.

APPENDIX A – Expert survey: V&V Methods and Quality Attributes.

We have selected some software Verification and Validation (V&V) methods and would like to have an initial understanding on how suitable they are to address different quality attributes. Based on your expertise you have been selected to help us gathering such initial understanding providing answers to two direct questions listed below. Answering the questionnaire will take around 20 minutes and your answers will be extremely valuable for our investigation.

At the end of the questionnaire, we provide a Glossary of terms as supporting documentation, which briefly describes each of the quality attributes and the V&V methods. In case of any further questions regarding this survey, please refer to Isela Mendoza (imendozadecgmail.com – M.Sc. Student) or Marcos Kalinowski (kalinowskiinf.puc-rio.br – Advisor).

Note: Individual data will be anonymized and the information will only be used for purposes related to the research.

Name (optional):

Question 1: To what extent do you agree that the following V&V methods (rows) can be applied to address the listed quality attributes (columns)?

1 - Disagree, 2 - Partially Disagree, 3 - Partially Agree, 4 - Agree, or N - Not Sure.

Classification	Methods \ Standards	Functional Suitability	Performance Efficiency	Compatibility	Usability	Reliability	Security	Maintainability	Portability
Based on Intuition & Experience	Ad Hoc								
	Exploratory Testing								
Input Domain-Based	Equivalence Partitioning								
	Pair wise Testing								
	Boundary-Value Analysis								
	Random Testing								
Code-Based	Cause-Effect Graphing								
	Control Flow-Based Criteria								
	Data Flow-Based Criteria								
Fault-Based	Error Guessing								
	Mutation Testing								
Usage-Based	Operational Profile								
	Usability Inspection Methods								
Model-Based Testing	Finite-State Machines								
	Workflow Models								
Reviews	Walkthrough								
	Peer Review or desk checking								
	Technical Review								
	Inspection								

