INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Localização automatizada de defeitos em C#

MICKAEL MAURICE LIMA SEBAN
Setembro de 2024

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Automated Fault Localization in C#

Mickael Seban

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Computer Systems

Supervisor: Dr. Alberto Sampaio, Professor, DEI/ISEP

Evaluation Committee:
President:
Dr. Alberto Sampaio, Professor, DEI/ISEP

Members:
Dr. Alberto Sampaio, Professor, DEI/ISEP

Porto, September 17, 2024

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, September 17, 2024

Dedicatory

I dedicate this dissertation to my beloved, Daniela, for her unconditional support, dedication, and love. Without her support, I would not have completed this journey. To my dear grandmother, Celeste, who raised me with such effort and always encouraged me to study. Her pride in my journey has been one of my greatest motivations.

Abstract

Debugging is one of the most challenging processes in software development. As the complexity of software increases, traditional debugging methods are becoming less viable and more expensive. Over the years, techniques for automatically locating defects, as well as improving them, and tools that implement these techniques have been developed. Although there have been significant improvements, the freely available tools still fall short when it comes to supporting C# projects.

This dissertation proposes the design and implementation of an automatic defect location tool for C# projects. The tool, developed during this dissertation, is called *FaultDetector.NET*. It contains two versions: one that integrates into Integrated Development Environment (IDE) *Visual Studio 2022*, and another that is a Command Line Interface (CLI) to support CI/CD pipelines. Some new defect localization techniques will be explored to bridge the gap between academic research and practical applications in software development.

A case study is carried out using the *Defects Sample Project*, a collection of C# projects containing reproducible bugs representative of different real-world software domains. Suspicion ratings and fault identification efficiency are used as metrics to compare the various fault localization techniques supported by the tool. This work focuses mainly on the importance of seamlessly integrating automatic fault localization into the development environments widely used by C# software engineers.

Keywords: Fault Localization, Fault Localization in C#, Automated Debugging, Fault Localization Techniques

Resumo

A depuração é um dos processos mais desafiantes no desenvolvimento de software. Com o aumento da complexidade do software, os métodos tradicionais de depuração são cada vez menos viáveis e mais dispendiosos. Ao longo dos anos, foram desenvolvidas técnicas para a localização automática de defeitos, bem como o seu aperfeiçoamento, e ferramentas que implementam essas técnicas. Embora tenham havido melhorias significativas, as ferramentas disponíveis gratuitamente ainda estão aquém, no que concerne a suporte a projectos C#.

Esta dissertação propõe a conceção e implementação de uma ferramenta de localização automática de defeitos em projectos C#. A ferramenta, desenvolvida durante esta dissertação, é denominada *FaultDetector.NET*. Contém duas versões: uma que se integra no IDE *Visual Studio 2022*, e outra que é um CLI para suportar os pipelines CI/CD. Algumas novas técnicas de localização de defeitos serão exploradas para colmatar a lacuna entre a investigação académica e as aplicações práticas no desenvolvimento de software.

Um estudo de caso é realizado usando o *Defects Sample Project*, uma coleção de projetos C# contendo bugs reproduzíveis representativos de diferentes domínios de software do mundo real. As classificações de suspeita e a eficiência da identificação de falhas são usadas como métricas para comparar as várias técnicas de localização de falhas suportadas pela ferramenta. Este trabalho centra-se principalmente na importância da integração perfeita da localização automática de falhas nos ambientes de desenvolvimento amplamente utilizados pelos engenheiros de software C#.

Acknowledgement

I also extend my thanks to my supervisor, Alberto Sampaio, for his guidance and availability throughout this journey.

Contents

List of Figures

List of Tables

List of Acronyms

AHP	Analytic Hierarchy Process.
CI/CD	Continuous Integration and Continuous Deployment.
CLI	Command Line Interface.
CS	Symmetry Coefficient.
DMB	Data Mining-Based.
DSRM	Design Science Research Methodology.
FEI	Front End of Innovation.
FFE	Fuzzy Front End.
IDE	Integrated Development Environment.
JSON	JavaScript Object Notation.
MBFL	Mutation-Based Fault Localization.
NCD	New Concept Development.
NCF	Number of Failed Test Cases that Cover a Statement.
NCS	Number of Successful Test Cases that Cover a Statement.
NF	Total Number of Failed Test Cases.
NPD	New Product Development.
NS	Total Number of Successful Test Cases.
NUF	Number of Failed Test Cases that Do Not Cover a Statement.
PSB	Program Slicing-Based.
SBFL	Spectrum-Based Fault Localization.
WPF	Windows Presentation Foundation.
XAML	eXtensible Application Markup Language.

Chapter 1

Introduction

This chapter outlines the context, problem, and objectives of this dissertation. Additionally, it briefly describes the expected results, performs a preliminary value analysis, and describes the methodology used. Finally, the structure of the document is outlined, providing an overview of the content of each of the following chapters.

1.1 Context

In software development, debugging is an integral part of the development cycle, but it is also very time-consuming, and this is exacerbated by increasing software complexity and size. Conventional debugging techniques, while helpful, are becoming less feasible with larger codebases. Indeed, studies indicate that almost half of the development effort is spent on fault debugging (Wen et al. 2019).

Automated debugging techniques embody an important paradigm change, which should minimize human interference and effectively point developers to faults. This dissertation aims to improve these automated techniques to master the still increasing complexity of software projects.

As promising as these approaches are, manual effort is still high. Particularly in C#, tools are not always that accessible for most developers. A proposed approach would be to improve fault localization in C# projects by integrating an existing tool or creating a new tool that integrates with the relevant Integrated Development Environment (IDE).

New and alternative techniques will be tested in this context. The applicability of the proposed solution will be tested on real-world code written in C#.

1.2 Problem

Debugging is a challenge in software engineering, itself acknowledged as complex and time-consuming: it is estimated that up to 50% of development time is spent on this activity because it is complicated and errors are hard to trace (Adragna 2008) (Parnin and Orso 2011).

Manually finding faults becomes even more challenging when software projects are large and complex. The research aims to address the problem of automating the debugging process to reduce the amount of time and effort spent. There are some tools available for automatic fault localization; however, they are not widely used. Improvements in this area have the potential to increase the efficiency and speed with which faults are located, as well

as democratize access to advanced tools for developers (W. E. Wong, Ruizhi Gao, et al. 2016a)

1.3 Objectives

The goal of this dissertation is to introduce an automatic fault localization tool within an IDE to enhance the current debugging methodology for C# codebases in the software development lifecycle. The objectives are outlined as follows:

1. **Identification and Analysis**: Develop or adapt an open-source automatic fault localization tool, fully integrated into the C# development environment (IDE), and explore new techniques and coefficients that could improve the accuracy and effectiveness of fault localization process. The overall goal is to provide a solution that improves usability and efficiency in the software development process, making fault localization in C# projects more effective.
2. **Selection and Evaluation**: Identify the most effective tools that supports integration with an IDE. This includes evaluating factors such as performance, usability, and compatibility with C# and the IDE.
3. **Integration and Enhancement**: Implement the selected design tools into a suitable IDE. This can maybe be improving the existent tools or create one from scratch.

1.4 Research Methodology

This research is anchored upon Design Science Research Methodology (DSRM), which can be augmented with critical ethnography to satisfy the requirement of this diverse world. The DSRM consists of six steps, as shown in Figure 1.1. These steps are: Problem identification and motivation, Definition of solution objectives, Design and development, Demonstration and efficacy, Evaluation of the solution's impact and outcomes, and lastly Communication of results and conclusion

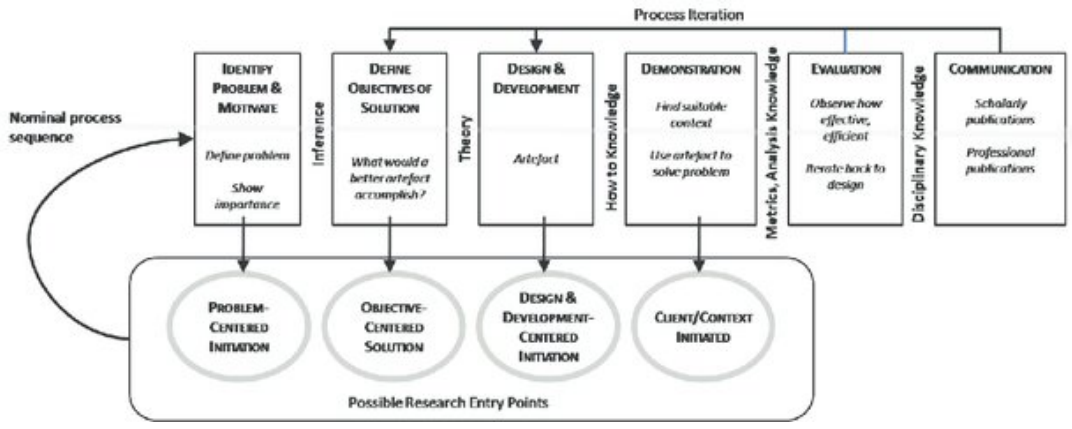


Figure 1.1: Design Science Research Methodology (Adapted from (Peffers et al. 2008)) - Extending Design Science Research Methodology for a Multicultural World

1.5 Methodology

The primary steps in this research are aligned with the Design Science Research Methodology (DSRM) framework, as follows:

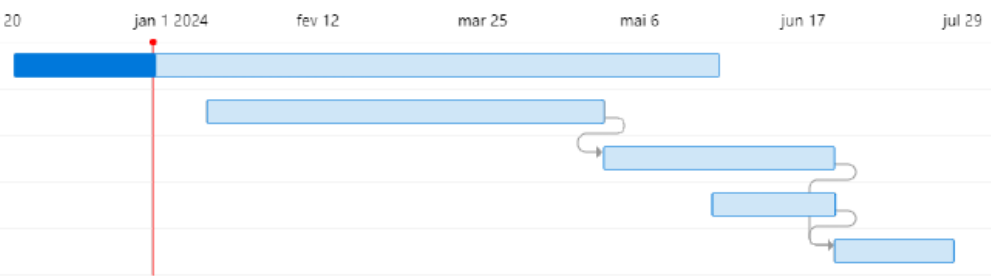
1. **Problem Identification and Motivation:** Recognize and define the core problems, particularly those exacerbated by contextual and cultural complexities. This motivation is grounded in improving the efficiency and effectiveness of debugging across diverse environments. *(See Chapter 1: Introduction)*
2. **Objective Definition:** Establish clear, actionable objectives based on identified problems. This includes designing IT artifacts that solve technical challenges while resonating with users from different cultural contexts. *(See Section 1.3)*
3. **Design:** Develop innovative solutions and artifacts to meet the defined objectives. This phase integrates insights from critical ethnography to ensure that cultural factors are incorporated into the design process. *(See Chapter 4)*
4. **Demonstration:** Implement and demonstrate the effectiveness of the designed artifacts in real-world settings, showing that the solutions address the identified problems and meet the objectives. *(See Chapter 5)*
5. **Evaluation:** Critically assess the artifacts using a combination of traditional DSRM evaluation methods and critical ethnography. This phase aims to understand the cultural impact and effectiveness of the solutions, ensuring they are beneficial in various contexts. *(See Chapter 5)*
6. **Communication:** Document and share the research findings, methods, and implications with both academic and industry communities. Additionally, this phase includes discussions on the extended DSRM methodology and its potential for broader applications. *(See Chapter 6)*

1.6 Work Plan

This section gives the 8-month timeline representing the major tasks involved in the work plan and their time allocation. These tasks include: State of the Art Study, Solution Design and Implementation, Experimentation and Evaluation, Solution Value Analysis, and Conclusion.

Automated Fault Localization in C#

	Nome	Atribuída a	Início	Conclusão	% Concluída
1	State-of-the-Art Review		1/12/2023	31/5/2024	20
2	Solution Design and Implementation		20/1/2024	1/5/2024	0
3	Experimentation and Evaluation		2/5/2024	30/6/2024	0
4	Solution Value Analysis		30/5/2024	30/6/2024	0
5	Conclusion		1/7/2024	31/7/2024	0



1.6.1 Detailed Work Plan

6 Months: State-of-the-Art Review

- Review existing literature on defect localization.
- Identify limitations in current solutions.
- Emphasize the need for further research.

4 Months: Design and Implementation

- Design a defect localization solution.
- Ensure it addresses existing limitations.
- Implement it in C#.

2 Months: Experimentation and Evaluation

- Conduct experiments to test solution effectiveness.
- Collect data and feedback for analysis.

1 Month: Value Analysis

- Assess the solution's value and impact.
- Evaluate practicality and effectiveness.
- Highlight benefits for software development.

1 Month: Conclusion

- Conclusion of the research and experiments;
- Generalization of the findings and discussion of its implications;
- The deficiencies of the prevailing defect localization tools are recognized.
- Systematization of knowledge through a literature review;
- Evaluation and consideration of variant defect localization approaches;
- Development and testing of the defect localization solution;
- Its efficiency is to be evaluated in a real software engineering context.

1.7 Document Structure

This document has the following structure:

1. **Introduction** - Chapter 1: This chapter sets the setting for the dissertation, which puts focus on the problem of automated defect localization. It describes the goals of the identification of current solution limitations, indication of the need for further research, and proposition of a novel approach. The foreseen results are more accessible and effective debugging, leading to large productivity gains in software engineering.

2. **State of the Art** - Chapter 2: The chapter surveys current automated debugging techniques, analyzing the existing work and its limitations. The author examine broad technical challenges, the lack of obvious solutions, and current inadequacies of the existing tools in support of these languages, particularly C#.

3. **Value Analysis** - Chapter 3: The chapter reassesses how the proposed work can add value for the customer and to the software engineering field. This includes elaborating on possible reductions in debugging time, gains in productivity, and perceived value from new product and process development. The analytic hierarchy process can also be covered here, considering ways of prioritizing improvements.

4. **Design and Implementation** - Chapter 4: This chapter highlights in detail the design and the implementation of the tool *FaultDetector.NET* from automatic fault localization in C# language.

5. **Evaluation and Experimentation** - Chapter 5: This chapter presents the evaluation and testing of the tool *FaultDetector.NET*. It includes experimentation using the *Defects Sample Project*, a collection of C# projects with reproducible bugs.

6. **Conclusion** - Chapter 6: The last chapter summarizes the research done, improvements achieved concerning fault localization for C#. Further, it discusses the limitations of the current solution and points out the work to be done in the future in order to make the tool's capabilities and broaden its applicability.

Chapter 2

State of Art

This chapter presents some concepts as well as the techniques and tools in the context of automated debugging. It references the Chapter 1 of the DSRM approach.

2.1 Debugging

2.1.1 Definition

Debugging is the process of identifying and correcting faults in computer software or systems in order to return the program to proper operation (Golagha et al. 2018a). These faults, commonly referred to as bugs, have made it essential to develop systematic approaches for their detection and repair.

There are five critical steps in a debugging process, as shown in Figure 2.1: identifying the failure, locating the fault, analyzing its nature, performing tests to validate the analysis, and covering any collateral effects to prevent future issues. Debugging is considered one of the most challenging and time-consuming tasks in software development, often requiring significantly more resources than the development itself (Srivastva and Dhir 2017); (Sayantini 2020)..

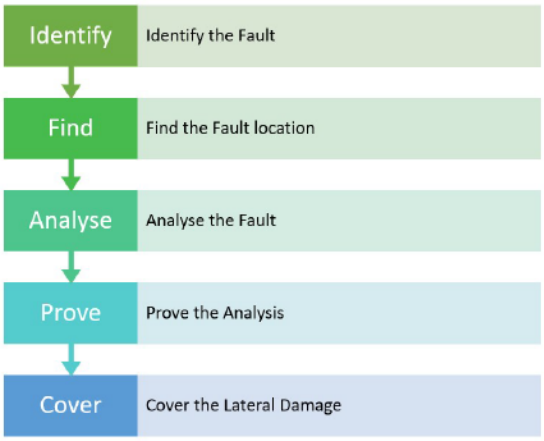


Figure 2.1: Debugging Steps (from (Sayantini 2020))

2.1.2 Debugging Tools

Debugging tools, or more simply debuggers, are special-purpose programs used to test and debug other software. They are key tools in mapping observed failures to their underlying

defects. These tools play a crucial role in helping software engineers build a mental model of the system, which can then guide the diagnosis process and ultimately lead to repairs.

Typical functionalities offered by these tools include a query processor, symbol resolver, expression interpreter, and a debug interface. Advanced features often include step-by-step execution of the program, halting at breakpoints to examine the state of the program at that point, and even the ability to set a different program state during execution.

The selection of a debugger is often determined by the operating system in use. For example, GDB is the standard for Unix/Linux, Visual Studio is commonly used for Windows, and LLDB for Mac OS. Each of these debuggers is designed to help engineers observe the dynamic behavior of the system, providing a finer-grained understanding and partial control over the debugging process, as also reported by Golagha et al. 2018a.

2.2 Error, Fault, and Failure

2.2.1 Definition

The terms "error", "fault", and "failure" are defined as follows: an "error" is a wrong system state; a "fault" is a defect in the system or its representation that might lead to an error in case of activation; and a "failure" is an observable deviation from the system's specification (International Organization for Standardization 2017).

The IEEE Standard (International Organization for Standardization 2017) also comments that "bug" is another synonym for fault, underlining that "a failure may be caused by (and thus indicate the presence of) a fault," and "a fault may cause one or more failures." Due to this fact, this thesis applies this terminology when referring to any computer program. In this sense, faults are bugs in the code, whereas failures are times when what is produced at the end differs from what the program is supposed to do.

Error detection is, however, a prerequisite for fault localization. One must realize that something is wrong before finding the cause of the fault. Failures are a primitive form of error detection but most errors go undetected and never lead to failures (Abreu, Pieter Zoeteweij, and Arjan J. C. van Gemund 2007). Detection of failure and array bounds checking are examples of generic error detection mechanisms that can be used without knowing the details of the system.

2.3 Fault Localization

When a software program encounters a failure, it is often due to multiple underlying errors within the code. To resolve these issues, it is essential to identify and correct the errors, which is the main goal of fault localization. This process involves analyzing various components of the source code to pinpoint the elements that may be responsible for the failure.

Fault localization typically requires examining many components and their interactions within the system. Due to the numerous potential causes of errors, automated fault localization techniques have been developed to help developers by reducing the complexity of error detection.

Fault localization techniques can be broadly classified into two categories: traditional methods, which are generally more straightforward, and advanced methods, which are more complex and sophisticated W. E. Wong, Ruizhi Gao, et al. 2016a. (W. E. Wong, Ran Gao, et al. 2016)

2.4 Traditional Methods

2.4.1 Program Logging

Program Logging is the process of capturing and recording all the details of the runtime Information on the program execution. Logging is used by the developer for monitoring variables, too. flow execution, and states of the system in order to understand the behavior of the program. and identify unusual or errant activities that signal a fault

2.4.2 Assertions

Assertions are concrete conditions explicitly stated in the code, according to which one is sure of the program behaving as expected, they act as checkpoints, and if an assertion fails, there might be a potential fault in the system. They help in early design error detection because the program can come to a stop if ever something goes wrong.

2.4.3 Breakpoints

Another definition of a breakpoint could be that it is a debugging tool allowing runtime suspension of execution at specific points in the program. That can let somebody look through the current state of computation, like variable values, call stack, or flow of execution. They are very instrumental in isolating and understanding the behavior of the code, in particular, near the suspected source of fault.

2.5 Advanced Methods

2.5.1 Program Spectrum-Based Fault

Spectrum-Based Fault Localization (SBFL) is one of the most prevalent techniques in identifying faulty components of a program during software debugging W. E. Wong, Ruizhi Gao, et al. 2016a. In essence, it means the analysis of program spectra, which are data sets collected during the execution of software and commonly referred to as code coverage data. This gives a fine-grained view into program behavior, based on tracing execution of various program entities, such as statements, branches, or methods while test cases are executed and can be collected by code instrumentation.

Program spectrum may be envisioned as a matrix comprising zeros and ones where every row corresponds to one test case and every column corresponds to some part of the program, for instance, concrete statement or branch. The entries of the matrix are 0 or 1 depending on whether a given test case executed a part of a program. Further, there is also an error vector which keeps track of whether each test case has failed or succeeded. (W. E. Wong, Ruizhi Gao, et al. 2016a)

Figure 2.2 shows an example of typical input to SBFL. Each row of the matrix represents to a test case, and each column corresponds to a part of the program (e.g., individual

statements or branches, the entries of the matrix x_{ij} indicate whether the j-th part of the program was executed during the i-th test case, a 1 in the matrix indicates that the part has been executed, and 0 indicates it was not.

$$\begin{array}{c} \\ M \text{ spectra} \\ \\ \\ \end{array} \begin{array}{c} \overset{N \text{ parts}}{\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ x_{21} & x_{22} & \dots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \dots & x_{MN} \end{bmatrix}} \\ s(1) \quad s(2) \quad \dots \quad s(N) \end{array} \quad \overset{\text{errors}}{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_M \end{bmatrix}}$$

Figure 2.2: Spectrum-Based Fault Localization Input Example (adapted from (Abreu, Peter Zoeteweij, Golsteijn, et al. 2009))

On the right-hand side of the matrix, an error vector **e** holds an outcome of each test case, where each entry e_i stands for either the i-th test case failedmostly represented as 1or successmostly represented as 0. SBFL compares each column vector $s(j)$ in the matrix with the error vector to identify which portion of the program is most likely responsible for the failures.

Ranking Suspicious Entities

Among the many dimensions of SBFL techniques, ranking the entities in a program according to their suspicion is one of the most important. SBFL techniques use various forms of ranking metrics and statistical methods to assign a higher priority to the parts of the program most likely to be faulty. The entities highly ranked are ones whose spectrum columns best match the error vector, thus most correlated with the failure of the test cases (Abreu, Pieter Zoeteweij, and Arjan J. C. van Gemund 2007).

Techniques

- **Tarantula**: It is among the earliest and most well-known techniques of SBFL. It ranks program entities based on their execution frequency in both passing and failing test cases. The suspiciousness score then falls between 0 and 1.

$$Suspiciousness = \frac{\frac{NCF}{NF}}{\frac{NCF}{NF} + \frac{NCS}{NS}}$$

- **Ochiai**: The Ochiai coefficient calculates the suspiciousness of a program entity in a very similar way to cosine similarity in vector space. The range for suspiciousness is [0, 1].

$$Suspiciousness = \frac{NCF}{\sqrt{NF \times (NCF + NCS)}}$$

- **Jaccard**: The Jaccard coefficient focuses on the proportion of failing test cases that execute a particular statement. The range of possible suspiciousness values is [0, 1].

$$Suspiciousness = \frac{NCF}{NCF + NUF + NCS}$$

- **DStar**: A family of techniques generalizing spectrum-based fault localization, introducing a tunable parameter; D3 is a popular variant. The range of possible suspiciousness values is $[0, +\infty)$.

$$Suspiciousness = \text{DStar} = \frac{(NCF)^3}{NCS + NUF}$$

Similarity Coefficients

Table 2.1 lists 31 similarity coefficient-based techniques, along with their algebraic forms, which have been used in different studies. Similarity coefficient-based techniques use mathematical formula to calculate the suspiciousness of each program entity based on the correlation between its execution in failing and passing test cases. Notable examples of these techniques include the use of the Jaccard coefficient, Tarantula, Ochiai, DStar, among others. Some tools, like Zoltar or DEPUTO, have been designed for computing those similarity metrics automatically and support fault localization (W. E. Wong, Ran Gao, et al. 2016; W. E. Wong, Ruizhi Gao, et al. 2016a).

Coefficient		Algebraic Form	Coefficient		Algebraic Form
1	Braun-Banquet	$\frac{N_{CF}}{max(N_{CF}+N_{CS}, N_{CF}+N_{UF})}$	17	Harmonic Mean	$\frac{(N_{CF}\times N_{US} - N_{UF}\times N_{CS})(N_{CF}+N_{CS})\times(N_{US}+N_{UF}) + (N_{CF}+N_{UF})\times(N_{CS}+N_{US}))}{(N_{CF}+N_{CS})\times(N_{US}+N_{UF})\times(N_{CF}+N_{UF})\times(N_{CS}+N_{US})}$
2	Dennis	$\frac{(N_{CF}\times N_{US}) - (N_{CS}\times N_{UF})}{\sqrt{n\times(N_{CF}+N_{CS})\times(N_{CF}+N_{UF})}}$	18	Rogot2	$\frac{1}{4}(\frac{N_{CF}}{N_{CF}+N_{CS}} + \frac{N_{CF}}{N_{CF}+N_{UF}} + \frac{N_{US}}{N_{US}+N_{CS}} + \frac{N_{US}}{N_{US}+N_{UF}})$
3	Mountford	$\frac{N_{CF}}{0.5\times((N_{CF}\times N_{CS}) + (N_{CF}\times N_{UF})) + (N_{CS}\times N_{UF})}$	19	Simple Matching	$\frac{N_{CF}+N_{US}}{N_{CF}+N_{CS}+N_{US}+N_{UF}}$
4	Fossum	$\frac{n\times(N_{CF}-0.5)^2}{(N_{CF}+N_{CS})\times(N_{CF}+N_{UF})}$	20	Rogers& Tanimoto	$\frac{N_{CF}+N_{US}}{N_{CF}+N_{US}+2(N_{UF}+N_{CS})}$
5	Pearson	$\frac{n\times((N_{CF}\times N_{US}) - (N_{CS}\times N_{UF}))^2}{N_C\times N_U\times N_S\times N_F}$	21	Hamming	$N_{CF}+N_{US}$
6	Gower	$\frac{N_{CF}+N_{US}}{\sqrt{N_F\times N_C\times N_U\times N_S}}$	22	Hamann	$\frac{N_{CF}+N_{US}-N_{UF}-N_{CS}}{N_{CF}+N_{UF}+N_{CS}+N_{US}}$
7	Michael	$\frac{4\times((N_{CF}\times N_{US}) - (N_{CS}\times N_{UF}))}{(N_{CF}+N_{US})^2 + (N_{CS}+N_{UF})^2}$	23	Sokal	$\frac{2(N_{CF}+N_{US})}{2(N_{CF}+N_{US}) + N_{UF}+N_{CS}}$
8	Pierce	$\frac{(N_{CF}\times N_{UF}) + (N_{UF}\times N_{CS})}{(N_{CF}\times N_{UF}) + (2\times(N_{UF}\times N_{US})) + (N_{CS}\times N_{US})}$	24	Scott	$\frac{4(N_{CF}\times N_{US} - N_{UF}\times N_{CS}) - (N_{UF}-N_{CS})^2}{(2N_{CF}+N_{UF}+N_{CS})(2N_{US}+N_{UF}+N_{CS})}$
9	Baroni-Urbani & Buser	$\frac{\sqrt{(N_{CF}\times N_{US})}+N_{CF}}{\sqrt{(N_{CF}\times N_{US})}+N_{CF}+N_{CS}+N_{UF}}$	25	Rogot1	$\frac{1}{2}(\frac{N_{CF}}{2N_{CF}+N_{UF}+N_{CS}} + \frac{N_{US}}{2N_{US}+N_{UF}+N_{CS}})$
10	Tarwid	$\frac{(n\times N_{CF}) - (N_F\times N_C)}{(n\times N_{CF}) + (N_F\times N_C)}$	26	Kulczynski	$\frac{N_{CF}}{N_{UF}+N_{CS}}$
11	Ample	$\left\lvert\frac{N_{CF}}{N_{CF}+N_{UF}} - \frac{N_{CS}}{N_{CS}+N_{US}}\right\rvert$	27	Anderberg	$\frac{N_{CF}}{N_{CF}+2(N_{UF}+N_{CS})}$
12	Phi (Geometric Mean)	$\frac{N_{CF}\times N_{US} - N_{UF}\times N_{CS}}{\sqrt{(N_{CF}+N_{CS})\times(N_{CF}+N_{UF})\times(N_{CS}+N_{US})\times(N_{UF}+N_{US})}}$	28	Dice	$\frac{2N_{CF}}{N_{CF}+N_{UF}+N_{CS}}$
13	Arithmetic Mean	$\frac{2(N_{CF}\times N_{US} - N_{UF}\times N_{CS})}{(N_{CF}+N_{CS})\times(N_{US}+N_{UF}) + (N_{CF}+N_{UF})\times(N_{CS}+N_{US})}$	29	Goodman	$\frac{2N_{CF}-N_{UF}-N_{CS}}{2N_{CF}+N_{UF}+N_{CS}}$
14	Cohen	$\frac{2(N_{CF}\times N_{US} - N_{UF}\times N_{CS})}{(N_{CF}+N_{CS})\times(N_{US}+N_{CS}) + (N_{CF}+N_{UF})\times(N_{UF}+N_{US})}$	30	Jaccard	$\frac{N_{CF}}{N_{CF}+N_{UF}+N_{CS}}$
15	Fleiss	$\frac{4(N_{CF}\times N_{US} - N_{UF}\times N_{CS}) - (N_{UF}-N_{CS})^2}{(2N_{CF}+N_{UF}+N_{CS}) + (2N_{US}+N_{UF}+N_{CS})}$	31	Sorensen-Dice	$\frac{2N_{CF}}{2N_{CF}+N_{UF}+N_{CS}}$
16	Zoltar	$\frac{N_{CF}}{N_{CF}+N_{UF}+N_{CS}+\frac{10000\times N_{UF}\times N_{CS}}{N_{CF}}}$			

Table 2.1: Similarity Coefficient-Based Techniques (from (W. E. Wong, Ruizhi Gao, et al. 2016a))

Adjustable Symmetry Coefficient

Measures, or coefficients, of similarity are commonly used in calculating the proximity between two entities are typically designed for symmetric or asymmetric data. Specifically, when comparing statements covered in passed tests to those uncovered in failed tests, the symmetry between these variables is nonexistent. In such cases, discordant pairs are usually ignored in the calculation of similarity between entities. A. Sampaio, I. B. Sampaio, and Alves 2007 suggest that it is possible to find a solution by identifying a parameter that adjusts the similarity measure according to the known level of asymmetry in the data .

The proposed solution involves finding a parameter that adjusts the similarity measure in proportion to the known amount of asymmetry. This is done by associating the Symmetry Coefficient (CS), as shown in Equation 2.1, with the uncovered statements in the failed tests (d), where n is the number of variables and NS is a parameter that varies proportionally to the global asymmetry present in the data A. Sampaio, I. B. Sampaio, and Alves 2007.

$$CS = \frac{n - NS}{n} \tag{2.1}$$

As A. Sampaio, I. B. Sampaio, and Alves 2007 suggests, this coefficient can be used in conjunction with well fault localization techniques. Tarantula

$$\text{Tar}_{CS} = \frac{\frac{NCF}{NF}}{\frac{NCF}{NF} + \frac{NCS}{NS}}$$

Simple Matching

$$\text{SM}_{CS} = \frac{(NCF + NCS) - \frac{n-NS}{n} \times (NCF + NCS)}{n + (NCF + NCS) - \frac{n-NS}{n} \times (NCF + NCS)}$$

Roger-Tanimoto

$$\text{RT}_{CS} = \frac{(NCF + NCS) - \frac{n-NS}{n} \times (NCF + NCS)}{(NCF + NCS) + 2 \times (NCF + NCS)}$$

Sokal-Sneath -

$$\text{SS}_{CS} = \frac{(NCF + NCS) - \frac{n-NS}{n} \times (NCF + NCS)}{(NCF + NCS) + 2 \times (NF + NS)}$$

Seban Technique

The author developed the technique to address the variability and limitations of individual SBFL techniques, such as *Tarantula*, *Ochiai*, *Jaccard*, *Kulczynski*, and *Rogers-Tanimoto*. Each of these techniques, although effective in specific contexts, tends to perform inconsistently depending on the nature of the program and the faults being diagnosed. This variability in effectiveness has been noted in several studies, including *"A Fault Localization Method Based on Metrics Combination"* by Ajibode et al. 2022, which emphasizes the need to combine metrics to improve precision and reliability in fault localization (Ajibode et al. 2022).

The Seban technique combines the results of each technique into a single suspicion metric to reduce the limitations of individual approaches and provide a more stable and accurate fault localization process. It integrates the following techniques: *Tarantula*, *Ochiai*, *Jaccard*, *Kulczynski*, and *Rogers-Tanimoto*.

The suspicion metric for the Seban technique is calculated as follows:

$$\begin{aligned}\text{Suspiciousness}_{\text{Seban}} = \text{Normalize}\Bigg(& \frac{\frac{\text{NCF}}{\text{NF}}}{\frac{\frac{\text{NCF}}{\text{NF}}}{\frac{\text{NCF}}{\text{NF}}+\frac{\text{NCS}}{\text{NS}}}} \\ & + \frac{\text{NCF}}{\sqrt{(\text{NCF}+\text{NUF})\times(\text{NCF}+\text{NCS})}} \\ & + \frac{\text{NCF}}{\text{NCF}+\text{NUF}+\text{NCS}} \\ & + \frac{1}{2}\left(\frac{\text{NCF}}{\text{NCF}+\text{NUF}}+\frac{\text{NCF}}{\text{NCF}+\text{NCS}}\right) \\ & + \frac{\text{NCF}+\text{NCS}}{\text{NCF}+\text{NCS}+2\times(\text{NUF}+\text{NUP})}\Bigg)\end{aligned}$$

A normalization step is included to ensure that the results fall within a standardized range between 0 and 1:

$$\text{Normalize}(x) = \frac{x}{\max(x)}$$

Where:

$$\max(x) = \max(\text{Suspiciousness}_{\text{Combined}})$$

The final stage of the process includes filtering to remove lines of code whose suspicion value is below the 90th percentile, thus helping developers focus on the most critical sections of the code:

$$\text{Suspiciousness}_{\text{Normalized}} < \text{PercentileThreshold}$$

Challenges and Ongoing Research

Recent studies have proposed new techniques and improvements to existing SBFL methods. For example, Ochiai2 extends the Ochiai formula by including additional factors that account for successful test cases that do not cover a statement. Its suspiciousness score is computed as follows:

$$Suspiciousness(Ochiai2) = \frac{NCF \times NUS}{(\text{NCF}+\text{NCS})\times(\text{NUS}+\text{NUF})\times(\text{NCF}+\text{NUF})\times(\text{NCP}+\text{NUS})}$$

Some of the other promising techniques are Nearest Neighbor technique which is based upon elements of fail a test case to the most similar passing test case. If the fault is found in the dierences between these tests, it is located; otherwise, a PDG is constructed to guide in a deeper investigation.

Despite their eectiveness SBFL techniques face several challenges, the major one is the overhead of instrumentation, is caused by the additional computational cost introduced by monitoring and recording the execution of the program during testing. Techniques like Coarse-grained instrumentation has been suggested to reduce this overhead, where the

program is first instrumented at a higher granularity, such as package or class level, and only the most suspicious components are instrumented more finely, such as at the statement level. Another challenge is how to ensure high accuracy of fault localization due to complex program structures and test suites. (W. E. Wong, Ruizhi Gao, et al. 2016a)

The quality and coverage of the test suite dominate the effectiveness of SBFL techniques. Incomplete test cases can result in inaccurate fault localization. Some research currently in focus is on improving these when multiple statements have the same score known by ties in suspiciousness scores and enhancing the precision in large and complex software's. Other recent studies frequently compare different spectrum- Whereas some spectrum-based fault localization techniques are available, none of them are arrogant enough to claim the best for all situations. That means there is no such spectrum-based technique in every situation that is optimal, as supported by the study of Yoo et al. 2014.

2.5.2 Slice-Based Techniques

The slice-based techniques are applied for abstracting a program by eliminating the irrelevant parts. Ensuring that the resulting slice has the same behavior as the original program with based on some specified criteria. Ever since the static slicing was introduced by Weiser 1979, the technique has been extensively studied and applied, particularly in the domain of fault localization, with hundreds of publications exploring its various dimensions (Gupta, Soffa, and Howard 1994; Harman and Hierons 2001; Tip 1995).

Static Slicing

Static slicing is mostly used for reducing the search space while debugging. The fundamental idea is that, given a failing test case, if the value of a variable is wrong at some statement, then the defect must lie within the static slice for that variable-statement pair. Thus, the developer can limit his debugging to a smaller region of the code (Lyle and Weiser 1987). Although it is an important tool, static slicing has its weaknesses, especially when there are pointer variables that can enable inefficient data flow analyses (Chandra, Reps, and Shankar 2007; Reps, Horwitz, and Sagiv 1995). Some techniques have been proposed, like equivalence analysis, to make data flow analyses ecient by computing equivalence relationships between memory locations.

Dynamic Slicing

A major limitation of static slicing is the fact that the slices generated may comprise all the executable statements that may have a possibility effect on a variable's value, hence incorporating many redundant statements within the debugging process. This is mainly because of the inability of static analysis to anticipate run-time values. Dynamic slicing is thus applied and it involves recognizing statements that truly contribute towards a particular value during a specific execution, thereby It more accurate for fault localization (Agrawal and J. Horgan 1993). However, dynamic slicing has its drawbacksone of which is that it cannot capture execution omission errorsscenarios where omission of crucial statements in the program causes failures (Gyimothy, Beszedes, and Forgacs 2001). To resolve this issues, relevant slicing and augmented dynamic dependence graphs have been proposed, although these methods might generate oversized slices due to incorrect dependencies (Xiangyu Zhang and Xu 2011).

Execution Slicing

An approach that is an alternative to both static and dynamic slicing, is the execution slicing, which uses dataow tests to construct slices and is based on statements which are executed by a particular test case (Agrawal and Joseph R. Horgan 1990). Being practically helpful, the execution slicing considers the behavior of the program under the actual inputs that expose the fault, and not under hypothetical inputs which could affect any variable of interest. The technique of this method is less resource-intensive and more easy to implement, especially with the availability of code coverage data (Jones, Harrold, and Stasko 2002).

Example: Comparing Static, Dynamic, and Execution Slicing

To differentiate between static, dynamic, and execution slicing, let us consider the code excerpt given in the second column of Table 2, which is infected with a bug at statement $s7$.

- The **static slice** for the output variable product consists of all statements that may inuence the value of *product*, hence it contains $s1, s2, s4, s5, s7, s8, s10$, and $s13$.
- The **dynamic slice** for *product* only contains those statements that actually inuence the value of *product* during a particular test case execution. Under input $a = 2$, the dynamic slice consists of $s1, s2, s5, s7$, and $s13$.
- The **execution slice**for the test case $a = 2$ is composed of all the statements executed by this test, hence it is composed of $s1, s2, s3, s4, s5, s6, s7, s12$, and $s13$.

Table 2.2 provides a graphical comparison between the outcome of these three slicing techniques. It allows to illustrate how the fact is that each of the methods captures different sets of statements depending on their criteria.

	Code with a bug at s_7
s_1	input(a)
s_2	$i = 1$;
s_3	$sum = 0$;
s_4	$product = 1$;
s_5	if ($i < a$){
s_6	$sum = sum + i$;
s_7	$product = product \times i$; //bug $product = product \times 2i$
s_8	}else{
s_9	$sum = sum - i$;
s_{10}	$product = product / i$;
s_{11}	}
s_{12}	print (sum);
s_{13}	print ($product$);

(a)

Static slice for $product$
input(a)
$i = 1$;
$product = 1$;
if ($i < a$){
$product = product \times i$;
}else{
$product = product / i$;
print ($product$);

(b)

Dynamic slice for $product$ with respect to a test case $a = 2$
input(a)
$i = 1$;
if ($i < a$){
$product = product \times i$;
print ($product$);

(c)

Execution slice for $product$ with respect to a test case $a = 2$
input(a)
$i = 1$;
$sum = 0$;
$product = 1$;
if ($i < a$){
$sum = sum + i$;
$product = product \times i$;
print (sum);
print ($product$);

(d)

Table 2.2: An Example Showing the Differences among Static, Dynamic, and Execution Slicing, adapted from (W. E. Wong, Ruizhi Gao, et al. 2016a)

While these techniques are useful, challenges still remain. For instance, there is absolutely no guarantee that the bug will be located within the generated slices, and even if the bug is located, the slices may still contain excessive code that needs to be examined. In order to enhance this, methods like inter-block data dependency augmentation and refining techniques have been proposed. More to that, program slices are of high complexity and big in length hence hard to understand. A barrier-based filtering technique and thin slicing were proposed for producing more manageable slices and making their comprehensibility higher (W. E. Wong, Ruizhi Gao, et al. 2016a).

2.5.3 Statistics-based Techniques

Statistical debugging is a technique that utilizes dynamic analysis to identify the underlying Reasons for software failures. In the past years, several algorithms have been proposed for improving This will ensure that the accuracy and efficiency of fault localization are effective. Some of the most Influential methods and their approaches.

Liblit05

The Liblit05 method, was introduced to isolate bugs in programs containing multiple undidiagnosed problems (Liblit et al. 2005). The technique emulates normal debugging process performed by programmers, where the most critical bug is identified first, fixed, and then the That process is repeated. It involves the use of instrumented predicates at specific It makes use of certain specific points in the program to predict the occurrence of failures

- **Feedback Report (R)**: It is the report that tells whether the execution of the program is successful or it has failed. If the predicate P is true at least once during the run, then$R(P) = 1$; otherwise, $R(P) = 0$
- **Bug Profile (B)**: A set of failing runs which share the same bug of failure.
- **Predicate (P)**: It is a bug predictor; when $R(P) = 1$, most probably it belongs to the bug profile.

This algorithm does its processing based on following step-by-step procedure:

1. Compute the probability that P implies failure, (*Failure(P)*).
2. Compute the probability that the execution of P implies failure (*Context(P)*).
3. Discard predicates where *Failure(P)* - *Context(P)* ≤ 0.
4. Rank the predicates which score the highest by their computed score, which is dependent on the importance which it provides according to the bug profile

According to Chilimbi and Hauswirth 2003 , the efficiency of Liblit05 could be enhanced by replacing the predicates with path profiles. These are runtime gathered and then aggregated from multiple test cases.

SOBER

The SOBER technique produced by Liu et al. 2006 ranks suspicious predicates by evaluating it within test cases. The technique calculates$\pi(P)$, which is the probability that predicate P is evaluated to be true during execution. A predicate P is related to if the distribution of $\pi(P)$ in failed executions has a higher signicantly different than from successful ones fault. J. Hu et al. 2010 made this much precise by using non-parametric hypothesis testing in checking the extent of variation between the successful and failed test cases spectra of predicates, hence obtaining more accurate fault localization. (W. E. Wong, Ruizhi Gao, et al. 2016b)

Crosstab

The Crosstab technique is based on cross-tabulation analysis to compute the suspiciousness of program statements (Xifeng Zhang et al. 2011). For each statement, a crosstab is constructed with two vertical categories (covered/not covered) and two horizontal categories (successful execution/failed execution). A hypothesis test is applied to assess the dependency between execution results and the coverage of each statement. Unlike Liblit05 Additionally, it can be used in general ranking of suspicious program elements: statements, predicates, and functions. It is shown that Crosstab really finds much more bugs than Liblit05 and SOBER. (W. E. Wong, Ruizhi Gao, et al. 2016b)

Debugging through Evaluation Sequences (DES)

Based on the identification of short-circuit evaluations in individual predicates and output evaluation sequences for each predicate, Z. Zhang et al. 2008 introduced the Debugging through Evaluation Sequences DES approach that was compared with the existing predicate-based techniques such as SOBER and Liblit05. You and You and Z. Zheng 2012 present a statistical approach that constructs for each test case a weighted execution graph with predicates as vertices and transitions between sequential predicates as edges. The suspiciousness of each edge is It is calculated to quantify its likelihood of being fault-related. Recent efforts applied causal-inference techniques to fault localization. A linear model is built on top of program control ow graphs to estimate the causal effect of covering a given statement on the occurrence of failures, reducing confounding bias and improving fault localization rankings. (W. E. Wong, Ruizhi Gao, et al. 2016b)

2.5.4 Model-Based Techniques

Model-Based techniques are based on identifying all possible causes of abnormal behavior in an observed system against the background of knowledge concerning the system's expected behavior under normal conditions. These approaches, therefore, anchor their analysis on the comparison between the model of the systemrepresenting expected behaviorand actual behavior observed during the execution of the program.

One of the pioneering works in model-based fault localization was presented by Maruyama and Matsuoka 2008, a generation of pre-fault models and anomaly detection based on these models are collected by the technique, which gathers function call traces to create an execution model. one that reflects the system behavior again. Suppose when the failure is detected, an anomaly score is computed. The average for each execution unit allows analysts to prioritize units with higher scores significantly. Reduces manual localization time. (W. E. Wong, Ruizhi Gao, et al. 2016b)

Another example is the dependency model-based technique by Mateis, Stumptner, and Wotawa 2000, which maps dependencies between statements in Java programs. It can also partly repair bugs. It supports object-oriented features, loop constructs, and global variables.

Techniques like that of Shchekotykhin, Schmitz, and Jannach 2016 incorporate partial One such relationship has, therefore, been attempts at formulating diagnoses to reduce the time required for sequential diagnosis sessions. These diagnoses are computed based on a subset of the minimal conflicts, forming a smaller search space and This includes, among others, ensuring that the actual cause of the problem, otherwise known as the preferred diagnosis, has been identified.

Abreu, Peter Zoeteweij, and Arjan J C van Gemund 2008 present a model-based technique at observation-level. This technique uses dynamic information, such as abstractions of program traces, in order to generate sub-models of the program under analysis. These sub-models are then used in order to compute valid diagnoses ranking them according to their likelihood of being responsible for the failures. Contrarily to most of the approaches, which usually This technique also includes a ranking with multiple fault explanations, even if it presents a single explanation. (W. E. Wong, Ruizhi Gao, et al. 2016b)

A critical concern with model-based techniques is how much power the model has in terms of expression. which greatly affects the success rate of the technique. Dependency-based

models, Models based on static/dynamic code analysis and abstraction-based models, which use Abstract interpretation to deal with complex structures like loops and recursive calls is a key ingredient. examples of such techniques (Wotawa 2002).

Model checking-based fault localization techniques make use of model checkers in the identification of bugs (Ball, Naik, and Rajamani 2004). If the model does not capture the program's specifications then the Model checker provides counter-examples which though doesn't directly indicate the faulty components, help narrow down the possible locations of causes for a bug (Groce and Visser 2006).

Sum up. the challenges to these techniques involve the need for accurate models and generalizing failure modes across different programs. That does not shut out model-based techniques; on the contrary, this is an area of research that remains very important and holds much potential for improving fault localization in accuracy and efficiency within complex systems.

2.5.5 Machine Learning-Based Techniques

Many successful applications of machine learning techniques in fault localization have been reported. It learns from statement coverage data and test case outcomes to infer the probable location of faults. The next section highlights three major machine learning-based fault localization approaches.

Back-Propagation Neural Networks

A technique for fault localization using BP neural networks is presented E. Wong and Qi 2016. BP neural networks turn into powerful function approximators of complex nonlinear functions. The training stage makes use of coverage data and execution results of test cases: either success or failure. Then, this network is used to rank all the statements of a program with respect to their suspiciousness. It is interpreted as the likelihood that the statement contains a bug. BP networks, though very powerful, are not immune to such problems as paralysis and local minima, which can degrade their performance.

Radial Basis Function Networks

To overcome the disadvantages of BP networks, E. Wong, Debroy, et al. 2012 propose a method using Radial Basis Function (RBF) networks. RBF networks exhibit faster learning and less sensitivity to paralysis and local minima. Training of the RBF network proceeds as for the BP network, but with three novel contributions: a new way to represent test cases and coverage information, an algorithm for estimating the number A hidden neuron and center, and a weighted bit comparison-based distance metric. These enhancements enable more accurate fault localization than possible with BP networks.

Deep Learning Techniques

(W. Zheng, Z. Hu, and J. Wang 2016) give an approach using Deep Neural Networks to localize faults. Deep learning is a subdomain of machine learning, in which models learn directly from large datasets like images, text, or sound. Concretely, the DNN is trained like other neural networks, with fewer examined statements: Test-suite reduction can identify all faulty versions of the software. Though very promising, this approach still needs improvements to handle multiple-bug programs and large-scale software systems.

2.5.6 Data Mining-Based Techniques

Data mining techniques, such as machine learning, are oriented to the discovery of patterns and correlations in large sets of data. Data mining in fault localization identifies patterns of statement execution that lead to failures. Some of the important data mining-based fault localization techniques are introduced in this section.

Association Rules and Formal Concept Analysis

Cellier et al. 2011, combine two data mining techniques: Association Rules and Formal. FCA): Concept Analysis. These methods work with binary relations describing executions by Subsets of the lines of programs and their outputs. This method is aware of rules that connect these It throttles executions down to program failures and builds a "failure lattice" that ranks the most suspicious. Program elements are ranked accordingly. After that, this ranking is used to guide fault localization.

Reinterpreting Tarantula as a Data Mining Problem

Denmat, Ducassé, and Ridoux 2005, redefines the well-known Tarantula technique as a data problem of mining. In this technique, data of the executed statements and the outcomes of test cases is then transformed for data mining purposes. Extraction and evaluation of association rules follow. It could also be measured with common data mining metrics like confidence and lift. These metrics help determine the likelihood that a statement contains a fault, offering an alternative perspective on the Tarantula method.

Fault Localization Framework Using Historical Data

Hirsch 2021 proposes a novel fault localization framework that leverages historical data, such as bug reports and commit messages. The framework uses a machine learning algorithm to create a model that assigns responsibility to code components. This approach inverts the traditional fault localization hierarchy by starting with historical data and then applying other fault localization techniques, offering a new avenue for improving fault localization efficiency.

2.5.7 Mutation-Based Techniques

Mutation-based fault localization techniques vary from the traditional methods most prominently in the use of mutation analysis as inputs to the ranking metrics. Below we now describe briefly two of the most notable mutation-based fault localization tools.

MUSE(Moon, Jung, and Bae 2014) and Metallaxis-FL(Papadakis and Le Traon 2015) are two of the most popular mutation-based fault localization tools. The main difference between the two tools is at the coefficient that they use while calculating the suspicion level score and the meaning of a killed mutant. For example, MUSE uses its own coefficient, whereas Metallaxis-FL uses the Ochiai coefficient. Moreover, MUSE considers a mutant as killed only when a failed test case changes to successful, Metallaxis-FL considers a mutant killed if the result of the test case is different, even if unsuccessful. These distinctions impact the effectiveness of the tools in finding faulty code.

2.5.8 Fault Localization Tools

Empirical studies in software fault localization often need the support of specialized tools for automatic or semi-automatic data collection and analysis of suspiciousness (W. E. Wong, Ruizhi Gao, et al. 2016a). In the following, some of the investigated tools are briefly introduced, and their purpose is pointed out.

Aletheia

Aletheia is a failure diagnosis toolchain targeted to assist developers and testers in reducing failure analysis time (Golagha et al. 2018b). Is available as a Visual Studio Extension and is particularly for analysis of C++ projects which have unit tests developed in the GTest framework. As Figure 2.3 depicts, the tool consists of three main components: data generation, failure clustering and fault localization. The data generation component is responsible for for the generation and preparation of data for subsequent analysis: running tests, gathering coverage information and aggregating it into spectra. The component Failure Clustering gets a hit spectrum as input and produces a hierarchical tree of failing tests, that are grouped into clusters by means of spectrum-based fault localization techniques. From these clusters, representatives are chosen based on the user's preferences. The last component is called Fault Localization: It uses the summarized hit spectrum to rank program elements by their likelihood of being faulty. Various fault localization techniques are supported, including Tarantula, Ochiai, Jaccard, and DStar.

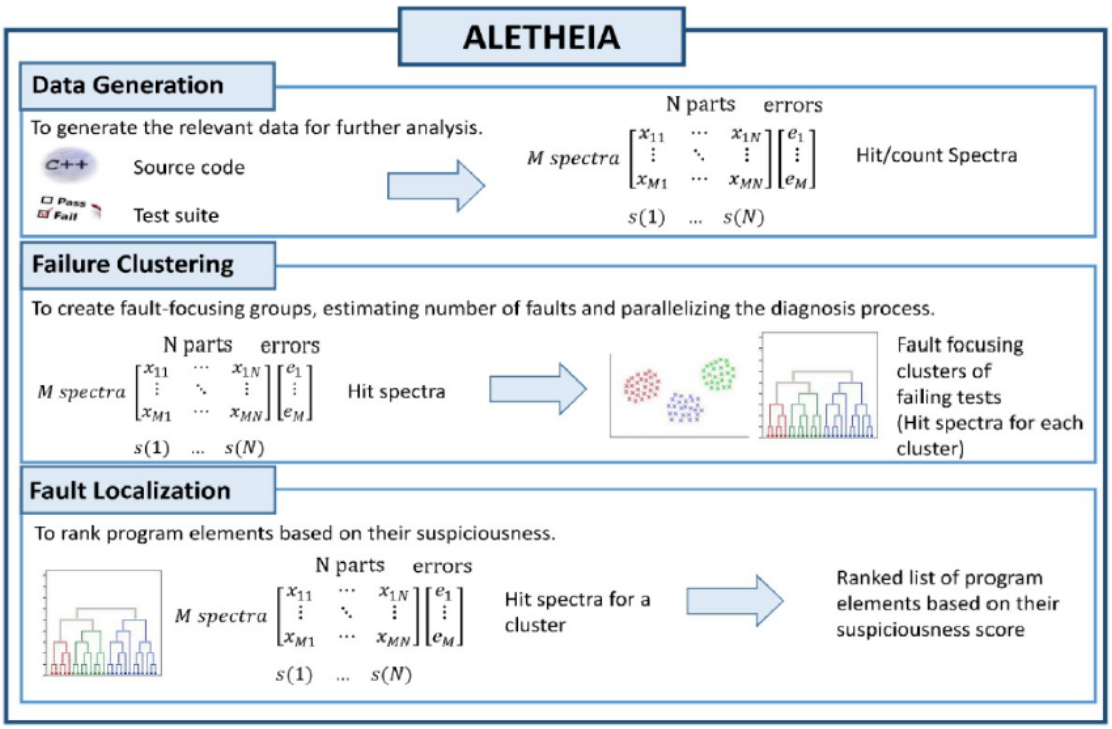


Figure 2.3: Aletheia main components from (Golagha et al. 2024)

CharmFL

CharmFL (Sarhan et al. 2021) is an open source fault localization tool which integrates with PyCharm, a popular Python development platform from Jetbrains. This tool uses

Spectrum-Based Fault Localization to help Python developers by evaluating the programs and generating data during runtime that can be used in order to generate a ranked list of potential faulty program elements. CharmFL can be divided into two major parts, the GUI and framework's architecture. In the GUI, program elements are highlighted with different shades of red depending on their suspicion level scoresthe darker the color, the more suspicious the element. The final output, as shown in Figure 2.4, is a hierarchical view of the program elements, along with their positions in source code, ranks and scores.

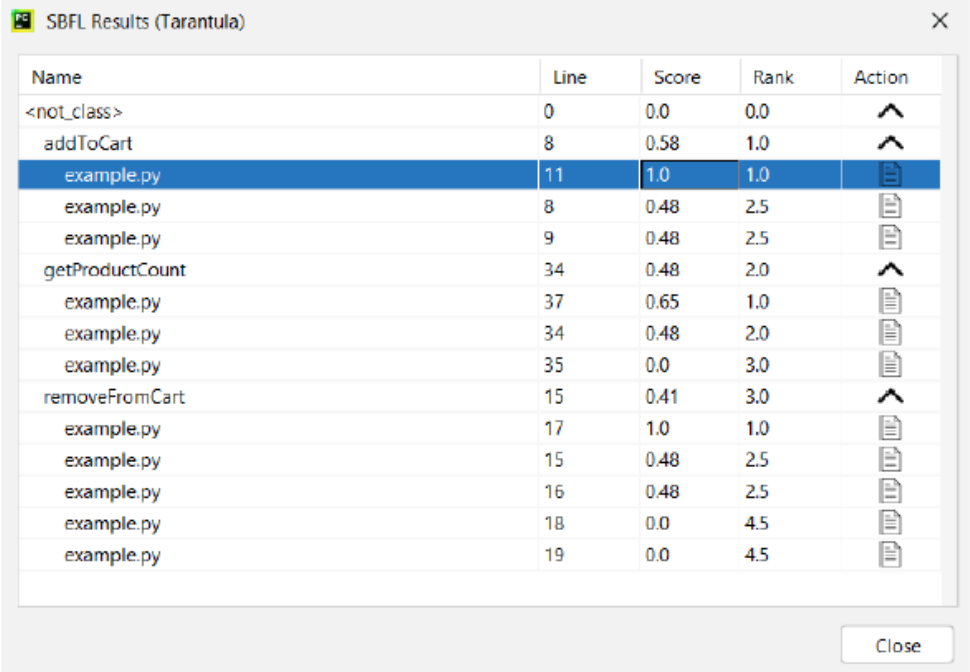


Figure 2.4: CharmFL ranking list output (adapted from Sarhan et al. 2021)

Regarding the frameworks architecture, it is important to note that CharmFL can be used as a standalone tool or integrated into other IDEs as a plugin. To collect the programs spectra, it uses a popular coverage measuring tool for Python called `coverage.py`. The framework transforms the statement-level data provided by `coverage.py` into method and class levels, as shown in Figure 2.4. The classes are then sorted based on their suspicion level scores, followed by the functions, and finally the statements. The coverage matrix is configured using the `.coveragerc` file, where the user can specify the measurement settings.

About the architecture of the framework, it is worth mentioning that *CharmFL* can operate as a standalone tool or as integrated into other IDE as a plugin. For gathering the program's spectra, it uses a well-known coverage measurement tool for Python called `coverage.py`. The framework translates the statement level data provided by `coverage.py` to method and class levels as depicted in Figure 2.4 The classes are then ranked based on their suspicion level scores are then followed by the functions, then finally the statements. The configuration of the coverage matrix happens in the `.coveragerc` file, where the user will be allowed to set measurement settings.

FLACOCO

FLACOCO (Silva et al. 2021) is a fault localization tool for Java. The tool is based on the SBFL technique and supports most versions of Java because of its base, JaCoCoone of the best code coverage librarieshaving been in the market for almost a decade. Adding support for new versions of Java in FLACOCO by mere upgrading of the JaCoCo version makes it comparatively easy and is an important improvement over other fault localization tools. Figure 2.5 shows the overall architecture of FLACOCO, which can roughly be divided into seven dif-ferent components: User Interface, Test Detection, Instrumentation and Test Execution, Coverage Collection, Suspiciousness Computation, AST Bridge and File Exporter. All of these components are explained briefly in the following, in this order:

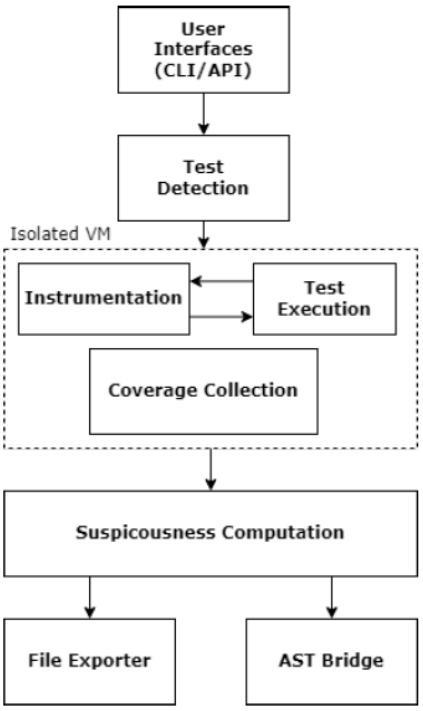


Figure 2.5: FLACOCO Architecture based on Silva et al. 2021

The first component represents the User Interface and, therefore, constitutes the entry point to the tool. It may be queried via a command line or as a Java API. The second component consists of Test Detection: it identifies the tests that have to be executed and then analyzed with the support of the following test frameworks: JUnit 3, 4, and 5. The third component involves the execution and instrumentation of the selected tests.JaCoCo. The Coverage Collection is the step following which data related to test results, thrown exceptions and code coverage is collected. On execution of the test suite, information collected is used to compute the suspicion level score using Ochiai coefficient. As per the paperSilva et al. 2021, the tool FLACOCO doesnot support IDE integration.

FLAVS

FLAVS stands for Fault Localization Add-in for Visual Studio, a tool to support developers in debugging and testing complex bugs in Microsoft Visual Studio (N. Wang et al. 2015). It logs various parameters during the software execution, and after a run, it allows the developers

to mark the status of the program. FLAVS then incrementally updates the suspiciousness of each statement, and highlights the most suspicious ones, thus making it easier for a developer to detect faults.

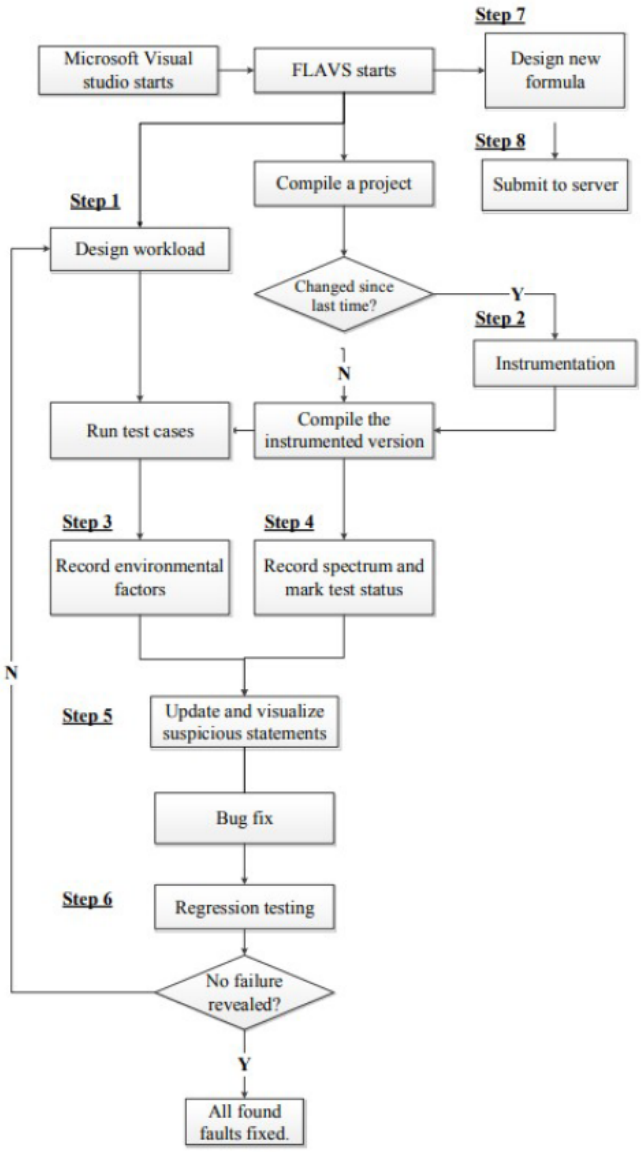


Figure 2.6: FLAVS flowchart (N. Wang et al. 2015)

GZoltar

GZoltar is a graphical user interface for debugging that takes as input Zoltar's output, a code dependency graph and a project hierarchy tree (J. Campos et al. 2012; Riboira and Abreu 2010). It supports a wide range of spectrum-based fault localization techniques. Ranking candidates based on collected data at runtime, it provides a ranked list of diagnosis candidates. As an ongoing open-source project, it is available in several forms, including command line interface, Ant tasks, Maven plug-in, and a Visual Studio Code Extension.

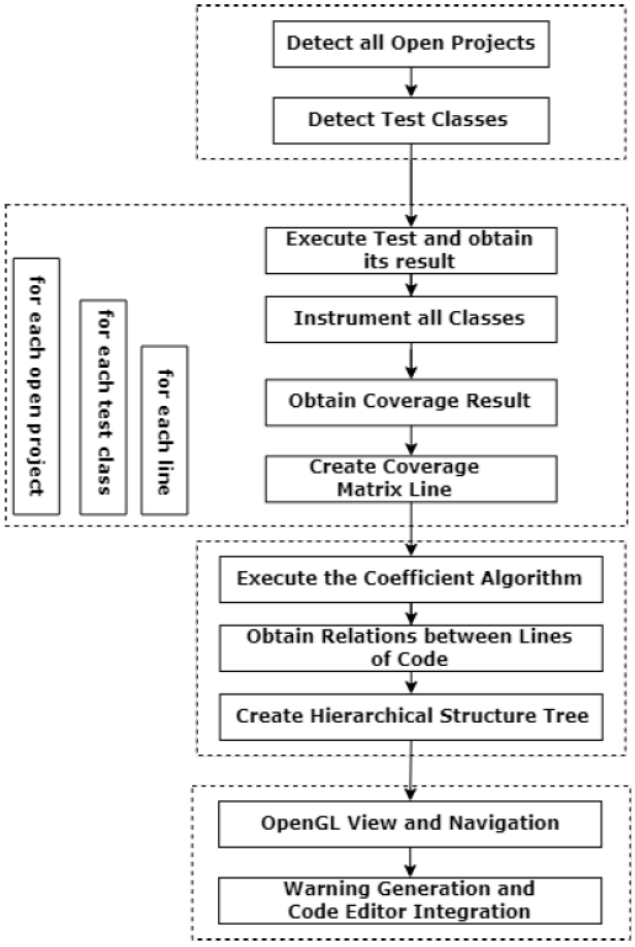


Figure 2.7: GZoltar information flow (J. Campos et al. 2012)

MZoltar

MZoltar is another GUI of a debugger, very close to GZoltar, but targeting fault detection. in mobile applications (Machado, J. C. Campos, and Abreu 2013). It utilizes the Android testing framework and JaCoCo to gather information on coverage, then provides a graphical It is an itemized form of the diagnostic report, which assists the developer in finding defects within the code.

Tarantula

Tarantula is a fault localization tool that employs the (SBFL) techniques and implements visual mapping (Jones, Harrold, and Stasko 2002). The tool It is written in Java and it visualizes a system according to various visual mappings based on source code and test-suite results, this helps in fault identification.

UnitFL

UnitFL (Chen and N. Wang 2016) is an add-in for Microsoft Visual Studio 2013 and supports projects written in C# and Visual Basic with tests written in NUnit 3.0 Testing Adapter. It operates in test mode and fault localization mode 2.8 and utilizes program slicing and

dynamic program instrumentation techniques to reduce program execution overhead. It evaluates each unit test's performance to uncover underlying bugs and guides the exploration of fault-related program units.

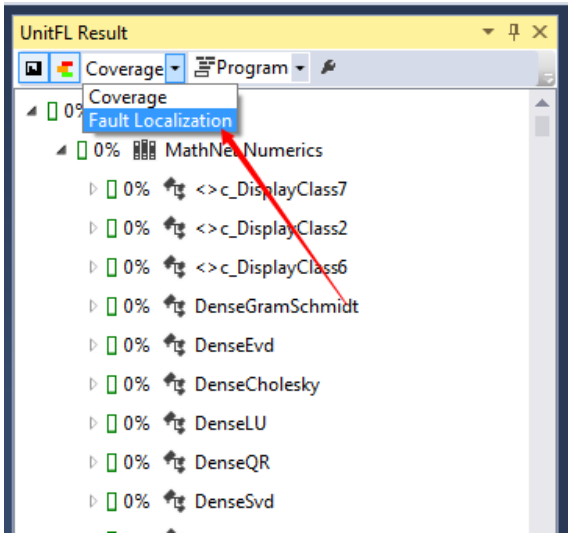


Figure 2.8: UnitFL Fault Localization mode (Wangnangg 2024)

After some tests fail, UnitFL calculates the suspiciousness of each program entity and ranks them in the result window 2.9. There are five levels of suspiciousness, represented by colors ranging from red to green. You can also navigate to the source code by clicking on an item, and you can toggle the coloration button to enable source code highlighting.

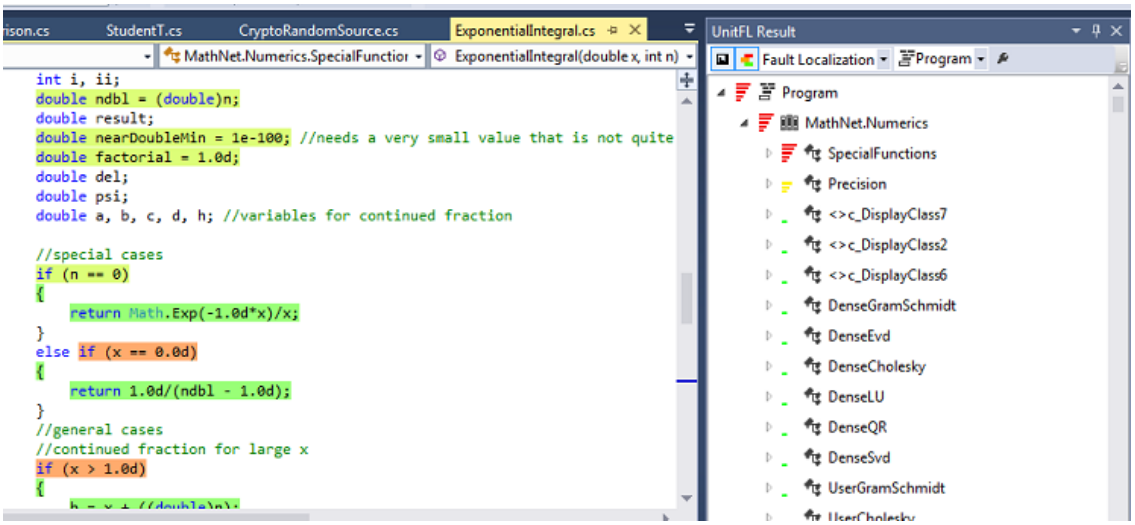


Figure 2.9: UnitFL Fault Localization result (Wangnangg 2024)

2.6 EXAM Score

As explained by Zou and al. 2021, the EXAM score quantifies the relative position of a faulty element within a ranked list of program statements. This score is calculated as $\frac{n}{N}$, where n

represents the rank of a defective statement, and N is the total number of statements in the program P. The score ranges from 0 to 1, with lower values indicating superior performance.

2.7 Comparative Analysis of Techniques

The key fault localization techniques were thoroughly discussed and explained in the section 2.5. However, a direct comparison between these families of techniques was not provided. To fill this gap, Zou and al. 2021 conducted a comparative evaluation using the EXAM score (See chapter 2.6).

This evaluation assesses the performance of the techniques on the *Defects4J* dataset, which at the time of study from Zou and al. 2021 had contains 357faults minimized from real-world faults in five open-source Java projects. While many tools from other techniques are not open source, this study focuses on comparing techniques such as SBFL (see Section 2.5.1), Mutation-Based Fault Localization (MBFL) (see Section 2.5.4), Program Slicing-Based (PSB) (see Section 2.5.2), and Data Mining-Based (DMB) (see Section 2.5.6).

Table 2.3: The Performance of Standalone Techniques on All 357 Faults (based on (Zou and al. 2021))

Family	Technique	@1	@3	$E_{inspect}$ @5	@10	EXAM
SBFL	Ochiai	16 (4%)	81 (23%)	111 (31%)	156 (44%)	0.033
	DStar	17 (5%)	84 (24%)	111 (31%)	155 (43%)	0.033
MBFL	Metallaxis	23 (6%)	78 (22%)	103 (29%)	129 (36%)	0.118
	MUSE	24 (7%)	44 (12%)	58 (16%)	68 (19%)	0.304
PSB	Union	5 (1%)	33 (9%)	58 (16%)	84 (24%)	0.207
	Intersection	5 (1%)	35 (10%)	55 (15%)	71 (20%)	0.222
	Frequency	6 (2%)	39 (11%)	58 (16%)	84 (24%)	0.208
DMB	BugLocator	0 (0%)	0 (0%)	0 (0%)	3 (1%)	0.212

According the table 2.3, the technique SBFL is the best-performing techniques. The Einspect is a measure that calculates the expected rank when multiple faulty elements are presented in ties. Based on these findings, it is recommended to prioritize techniques that utilize SBFL, as they demonstrate superior EXAM scores and are supported by more current and relevant information.

2.8 Comparative Analysis of Tools

Table 2.4: Comparative Analysis of Fault Localization Tools

Tool	Technique	IDE Integration	Language Supported	Open Source	Year
Aletheia	SBFL	Visual Studio	C++	Yes	2018
CharmFL	SBFL	PyCharm	Python	Yes	2021
FLACOCO	SBFL	None	Java	Yes	2021
FLAVS	SBFL	Visual Studio	C/Visual Basic/C#	No	2015
GZoltar	SBFL	Visual Studio Code	Java	Yes	2012
MZoltar	MBFL	None	Multi language	No	2015
Tarantula	SBFL	None	Java	No	2002
UnitFL	Slicing	Visual Studio	C#/C/Visual Basic	No	2016

2.9 Conclusion

In this work, the key fault localization techniques were thoroughly discussed and explained (see Section 2.5), along with the most relevant fault localization tools (see Section 2.5.8). A comparative analysis of both techniques and tools was conducted (see Section 2.7 and Section 2.8), allowing us to determine that the best-performing technique is SBFL. However, it was also concluded that there is no existing tool that simultaneously supports the C# language, is open-source, and implements SBFL.

Chapter 3

Value Analysis

This chapter addresses automated debugging potential and it affects its value as a technique.

3.1 Business and Innovation Process

The Business and Innovation Process is the major and most important stage of development for new products. frameworks, or ideas. The process, according to to P. Koen et al. 2001, may be broken down into three Distinct Parts: Fuzzy Front End (FFE) or Front End of Innovation (FEI), New Product Development (NPD), and commercialization. The understanding of these phases, all are critical to manage effectively to make an innovation successful.

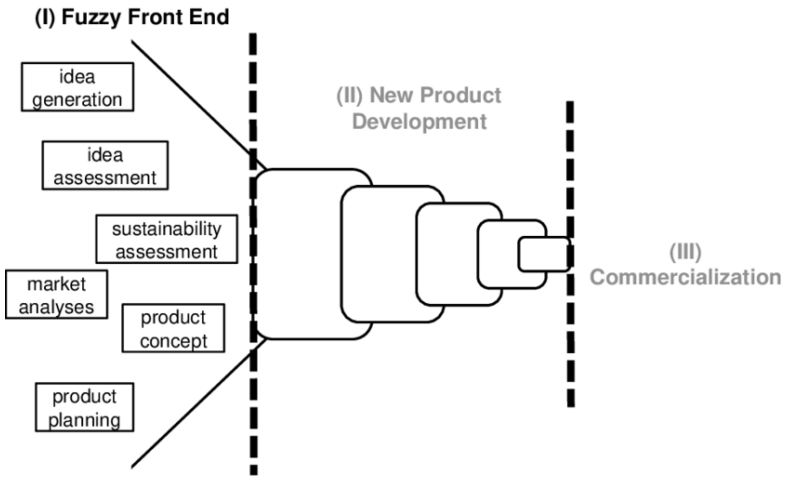


Figure 3.1: The three main steps of the innovation process (from (P. Koen et al. 2001)) and (Herstatt and Verworn 2001))

3.1.1 Fuzzy Front End or Front End of Innovation The Fuzzy Front End or FEI

It represents the very first process of innovation. The activities in this phase are fuzzy and lack formal structure. This phase consists of idea generation, opportunity identification, and concept development. During the FFE, all kinds of possibilities are explored, and some insights are gained that might lead to possible innovations. This process is very important because it sets the base for any further development and has to be open-minded and creative to detect feasible new ideas.

3.1.2 New Product Development (NPD)

When ideas with promising potential have been generated and developed into feasible concepts, these concepts enter the New Product Development (NPD) stage. This stage is more formalized and the first concepts are elaborated to physical products or services. NPD is a series of steps that involve designing, prototyping, testing, and iterating. The goal of this phase is to perfect the product, uncover any possible problems, and to confirm if the product fits the intended specifications and meets the needs of the market. Good NPD processes are key to reducing risks and enhancing the chances of success in product launch.

3.1.3 Commercialization

The last stage of the innovation process is Commercialization. This This phase intends to expose the developed product to the market and make it available to customers. Commercialization entails marketing, sales, distribution, and post-launch support. The sole aim of this stage is to achieve large diffusion while generating revenue streams. Successful commercialization will have to address wide knowledge of the targeted market, with effective marketing techniques and well-built support systems for customer satisfaction and loyalty. For businesses, the process of innovation goes through three important stages: Fuzzy Front End or Front End of Innovation, New Product Development, and Commercialization. Each phase plays a very significant role in the development of successful products from ideas. Distinct strategies and approaches would be required to move through these phases. Mastering these alone can any organization develop its innovation capability to achieve long-term growth.

The *New Concept Development New Concept Development (NCD)* model is a framework for the Front End of Innovation (FEI) that provides a common knowledge base for the field. This model is divided into three fundamental parts:

1. **The Engine**: At the center of the model is the engine, which provides power to the innovation process. The engine consists of two separate segments:
    - **Organizational Attributes/Teams**: These are the characteristics and capabilities of the teams involved in the innovation process.
    - **Collaboration**: This segment emphasizes the importance of teamwork and cooperation among different members and departments within the organization to drive innovation.
2. **The Wheel**: The inner part of the model is represented by the wheel. This component contains the five fundamental elements of the front end of innovation:
    - **Opportunity Identification**: The process of recognizing potential areas for innovation.
    - **Opportunity Analysis**: Evaluating the identified opportunities to determine their feasibility and potential impact.
    - **Idea Generation**: Creating new ideas based on the identified and analyzed opportunities.
    - **Idea Selection**: Choosing the most promising ideas from the pool of generated concepts.

- **Concept Definition**: Developing the selected ideas into detailed concepts that are ready for the development phase.

3. **The Rim**: The outermost part of the model is the rim, which comprises the environmental factors that influence the engine and shape the five activity elements. These factors include:
   - Market trends
   - Technological advancements
   - Regulatory conditions
   - Competitive dynamics

   The rim ensures that the innovation process remains responsive to external influences and aligned with the broader context in which the organization operates.

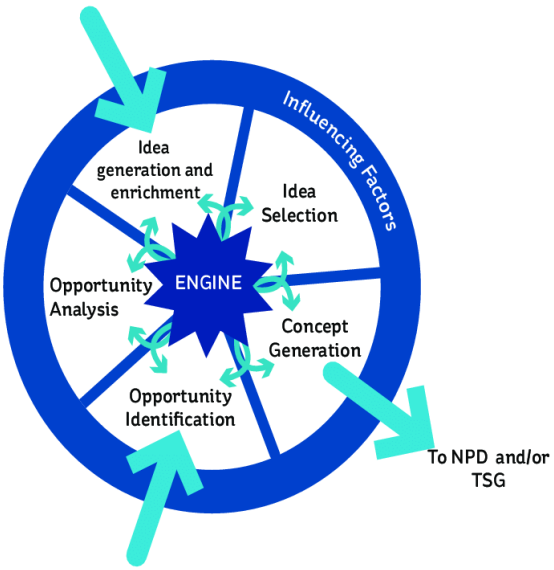


Figure 3.2: The New Concept Development Model (NCD)(from (P. Koen et al. 2001))

3.1.4 Opportunity Identification

Opportunity identification is where an organization identifies various opportunities to use its resources effectively. This may involve optimizing existing processes to make them more efficient or cost-effective, or it could include exploring entirely new directions that the business has proposed to venture into, such as developing a new service or a manufacturing process (P. A. Koen and al. 2002).

In the context of this dissertation, the opportunity identification lies in the field of fault localization, specifically in supporting the C# language. It is widely accepted that debugging is a notoriously difficult, human-intensive activity that is extremely time-consuming. According to (Wen et al. 2019), this process can consume up to 50% of the development and maintenance effort.

As a result, several automated techniques and tools have been developed to support various debugging tasks. However, despite their potential, many of these techniques have not yet

proven to be practically effective (Parnin and Orso 2011). Many approaches make strong assumptions about developer behavior during debugging, focusing primarily on reducing the number of statements to inspect, while often overlooking the importance of identifying relevant inputs.

Therefore, the challenges in debugging represent a significant opportunity for improvement in automated debugging techniques, which can help reduce the burden of debugging activities. Enhancements in this domain can improve productivity and the overall performance of the software development cycle. This opportunity is amplified by the absence of an open-source fault localization tool that supports the latest versions of C# and integrates seamlessly with popular IDEs to ensure availability and usability for developers.

Opportunity Analysis

Once an opportunity has been identified, it needs to be checked based on its potential, so that attention can be paid to whether the investment is justified by the effort. Recently, the concept of "Fault Localization in C# code" has attracted considerable attention, and various techniques are emerging that will transform this concept into a practical solution. These techniques hold great promise for improving the debugging process and will go a long way in aiding the development of high-reliability and quality applications in shorter timeframes. This dissertation incorporates an initial assessment of available techniques and their potential impact, with a focus on those discussed and presented in Chapter 2.

3.1.5 Idea Generation and Enrichment

Idea generation is the evolutionary process that includes the birth, development, and maturation of an opportunity into a definitive idea (P. A. Koen and al. 2002). This process often involves multiple iterations, reshaping, combining, modifying, or upgrading initial concepts. Brainstorming sessions are most useful for acquiring a variety of ideas and drawing inspiration from them. The initial idea in this dissertation was proposed by Prof. Dr. Alberto Sampaio. After some brainstorming sessions, the idea was refined into either incorporating an existing fault localization tool for the C# language into an IDE as an add-in or developing a completely new one from scratch. Besides the main idea, there are still some other questions to answer:

- **Q1:** What fault localization techniques should the tool implement?
- **Q2:** Which tool will be used?
- **Q3:** Which IDE will support integration with the tool?

The main focus was on Q1, because the choice of fault localization technique will define the entire approach. After thorough investigation, the following five alternatives were identified:

- **A1:** Spectrum-Based Fault Localization
- **A2:** Program Slicing-Based
- **A3:** Mutation-Based Fault Localization
- **A4:** Data Mining-Based
- **A5:** Machine Learning-Based

3.1.6 Idea Selection

The selection of an idea is a crucial and challenging process, and proper idea selection is vital for maximizing the value derived from an opportunity. Though the process might be as simple as selecting one idea from a set of self-generated options, it can prove to be a critical success or failure factor. Even though the selected idea can further evolve and develop over time, it must be based on a sound foundation (P. A. Koen and al. 2002). Moreover, no system can guarantee that the right decision will be made.

However, several techniques can improve the idea selection process, increasing the chances of success. One of these techniques is the Analytic Hierarchy Process (AHP), which was employed for this purpose in this thesis.

3.2 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) is a technique designed by Thomas Saaty, a mathematician, that introduced prioritizing options for the first time in the 1970s. Its major application is to provide solutions in decision-making and estimation problems in multivariate environments. AHP establishes priority weights for alternatives by organizing objectives, criteria, and sub-criteria into a hierarchy (Bernasconi, Choirat, and Seri 2009).

To make a decision in an organized way and generate priorities, the decision-making process should be decomposed into four steps (Saaty 2008):

1. Define the problem and determine the type of knowledge required.
2. Structure the decision hierarchy starting from the top with the goal of the decision, flowing through the objectives from a broad perspective, through the middle levels (criteria on which other elements depend), to the lowest level (which is usually a set of alternatives).
3. Construct a set of pairwise comparison matrices. Each element in an upper level is used to compare the elements in the level immediately below with respect to the concern.
4. Use the priorities obtained from the comparisons to weight the priorities in the level immediately below. Repeat this for all elements. For each element in the lower level, add its weighted values and get its overall or global priority. Continue this process of weighting and adding until the final priorities of the alternatives at the lowest level are obtained.

This section applies AHP in order to find the optimal techniques of fault localization based on the following criteria:

- **Accuracy**: How many faults it is able to detect.
- **Reliability**: How predictable its performance is.
- **Performance**: How long it takes for the technique to locate the faults and present them as a result.

The alternative techniques are:

- Spectrum Based

- Program Slicing
- Mutation Based
- Data Mining Based

The generated AHP can be seen below in Figure 3.3:

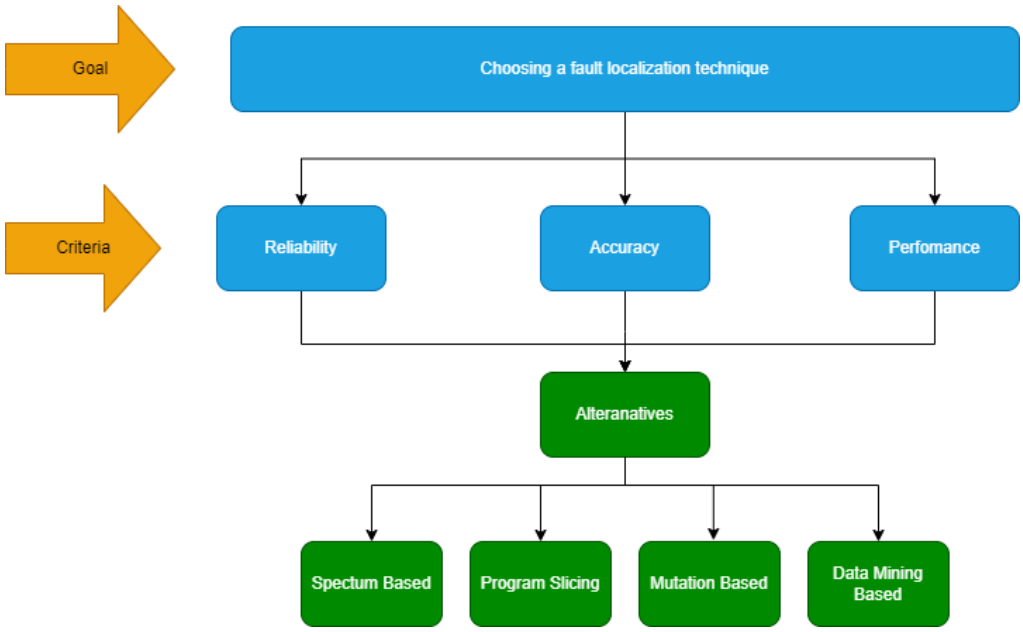


Figure 3.3: AHP Hierarchical Decision Tree

3.2.1 Pairwise Comparisons

Each rectangle in the hierarchy illustrated in Figure 3.3 represents a node, which is evaluated based on pairwise comparisons among all other nodes.

Next, finish step 3 by comparing each level pair by pair in terms of their contribution to the nodes just above these levels, directly in sequence. Fill out the matrix with results for the purpose of providing priorities to all the nodes at the head level of detail.

Now, five alternatives for fault localization techniques are considered: Spectrum Based, Program Slicing, Mutation Based, and Data Mining Based. Each pair of techniques is then compared based on each criterion. The specific comparisons that should be made are as follows:

- **Spectrum Based vs. Program Slicing**
- **Spectrum Based vs. Mutation Based**
- **Spectrum Based vs. Data Mining Based**
- **Program Slicing vs. Mutation Based**
- **Program Slicing vs. Data Mining Based**
- **Mutation Based vs. Data Mining Based**

For each criterion, such as accuracy, reliability, or performance, comparisons are made for each of these pairs. In every comparison of the pair, the technique that is weaker based on the criterion being evaluated is identified first. Then, a relative weight is assigned to the other fault localization technique.

The AHP fundamental scale for pairwise comparisons is provided in Table 3.1.

Table 3.1: The fundamental scale for pairwise comparisons (Saaty 2008)

Intensity of Importance	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the same objective
2	Weak or slight	Experience and judgment slightly favor one activity over another
3	Moderate importance	Experience and judgment slightly favor one activity over another
4	Moderate plus	Experience and judgment strongly favor one activity over another
5	Strong importance	Experience and judgment strongly favor one activity over another
6	Strong plus	An activity is favored very strongly over another; its dominance demonstrated in practice
7	Very strong or demonstrated importance	An activity is favored very strongly over another; its dominance demonstrated in practice
8	Very, very strong	The evidence favoring one activity over another is of the highest possible order of affirmation
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation
Reciprocals of above		If activity i has one of the above non-zero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i
1.1-1.9	If activities are very close	May be difficult to assign the best value but when compared with other contrasting activities the size of small numbers would not be too noticeable yet they can still indicate the relative importance of the activities

Ultimately, by utilizing the AHP fundamental scale mentioned in Table 3.1, weights are assigned to each criterion in relation to the other automated debugging techniques. Furthermore, each criterion is subjected to pairwise comparisons. For example, performance is compared against both accuracy and reliability. This same procedure is applied consistently across all criteria.

Table 3.2: AHP criteria comparison

	Accuracy	**Reliability**	**Performance**
Accuracy	1	2	7
Reliability	1/2	1	5
Performance	1/7	1/5	1

By analyzing Table 3.2, we can conclude that:

- Accuracy is slightly more important than reliability.
- Accuracy is very strongly more important than performance.
- Reliability is strongly more important than performance.

In this case, accuracy is the most determining factor in an algorithm, especially in fault localization since the goal is to discover as many faults as possible. This is followed by reliability because, in order for it to be a good technique, it needs to be consistent. Lastly, performance is a useful metric to have since it can help decide between two really good options, but it is not as critical as the previous two.

Criterion Relative Priority

Now, there are two important steps to do in the third step. First, in order to match All of the criteria will be transformed into the same unit, which means the comparison matrix values are normalized. To do this, we must find the sum of each column.

Table 3.3: AHP table for criteria comparison

	Accuracy	**Reliability**	**Performance**
Accuracy	1	2	7
Reliability	1/2	1	5
Performance	1/7	1/5	1
Sum	1.7143	3.2	13

And then, normalize by dividing each matrix value by the sum of its associated column

Table 3.4: Normalized criteria matrix

	Accuracy	**Reliability**	**Performance**
Accuracy	0.5833	0.625	0.5385
Reliability	0.2917	0.3125	0.3846
Performance	0.1250	0.0625	0.0769

The priority vector needs to be obtained, in the end, to establish the order of relevance of every criterion. To this end, it makes use of the normalized matrix from the previous point to find the arithmetic mean for the values of every row.

Table 3.5: Relative Priority of each criterion

	Accuracy	**Reliability**	**Performance**	**Relative Priority**
Accuracy	0.5833	0.625	0.5385	0.5823
Reliability	0.2917	0.3125	0.3846	0.3296
Performance	0.1250	0.0625	0.0769	0.0881

Relative Priorities Coherence

The fourth step now, in quantifying how consistent the judgments were for large samples of completely random judgments, is to calculate the Consistency Ratio (CR). If the CR exceeds 0.1, the judgments cannot be trusted as they are too close to randomness.

First, find the value of λ_{max} using the matrix A and the largest eigenvalue as its solution:

$$Ax = \lambda_{max}x$$

Where x is the priority vector of each criterion, and Ax is the multiplication of this vector by the non-normalized priority matrix. The matrix below shows the results of this multiplication.

$$\begin{pmatrix} 1 & 2 & 7 \\ 1/2 & 1 & 5 \\ 1/7 & 1/5 & 1 \end{pmatrix} \times \begin{pmatrix} 0.5823 \\ 0.3296 \\ 0.0881 \end{pmatrix} = \begin{pmatrix} 1.7452 \\ 0.9228 \\ 0.2748 \end{pmatrix}$$

Figure 3.4: Multiplication of priority vector by the non-normalized priority matrix

Then, it is possible to find the matrix eigenvalue:

$$\lambda_{max} = \frac{1.7452}{0.5823} + \frac{0.9228}{0.3296} + \frac{0.2748}{0.0881} \approx 3.0007$$

Finally, the gap between the Consistency Index (CI) and the Random Index (RI) is required to deduce the Consistency Ratio (CR). The formula to calculate the CI is:

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

Where n is the number of criteria involved. Given that the value of λ_{max} has already been determined, the CI can be inferred quickly:

$$CI = \frac{3.0007 - 3}{3 - 1} \approx 0.00035$$

The Consistency Ratio (CR) is then obtained as:

$$CR = \frac{CI}{RI}$$

Assuming $RI = 0.58$ for three criteria, which is quite common in most AHP usages, we can obtain:

$$CR = \frac{0.00035}{0.58} \approx 0.0006$$

As the CR is much less than 0.1, the judgments are consistent.

Alternative's Pairwise Comparison Matrix For Each Criterion

Step Five: Contemplate the relative relevance of each of the alternatives vis-à-vis the problem in question. Indeed, to accomplish this, operations are repeated from the creation of comparison matrices, this time between each of the criteria and alternatives, the respective normalization matrix, and the calculation of the priority vector. Tables 3.6, 3.7, and 3.8 show the preliminary comparison done using the fundamental scale.

Table 3.6: Comparison Matrix of Accuracy between Alternatives

Accuracy	SFL Based	Program Slicing Based	Mutation Based	Data Mining Based
SFL Based	1	3	4	7
Program Slicing Based	1/3	1	5	5
Mutation Based	1/4	1/5	1	3
Data Mining Based	1/7	1/5	1/3	1

Table 3.7: Comparison Matrix of Reliability between Alternatives

Reliability	SFL Based	Program Slicing Based	Mutation Based	Data Mining Based
SFL Based	1	4	5	8
Program Slicing Based	1/4	1	2	6
Mutation Based	1/5	1/2	1	4
Data Mining Based	1/8	1/6	1/4	1

Table 3.8: Comparison Matrix of Performance between Alternatives

Performance	SFL Based	Program Slicing Based	Mutation Based	Data Mining Based
SFL Based	1	2	3	5
Program Slicing Based	1/2	1	2	4
Mutation Based	1/3	1/2	1	3
Data Mining Based	1/5	1/4	1/3	1

These matrices are then normalized, and the local priority vector for each one of them is determined. Table 3.6, 3.7, and 3.8 show the results as well as their local priority vector.

Table 3.9: Normalized Matrix for Alternatives-Accuracy Comparison and Local Priority

Accuracy	SFL Based	Program Slicing Based	Mutation Based	Data Mining Based	Local Priority
SFL Based	0.6231	0.6848	0.7463	0.3889	0.5615
Program Slicing Based	0.2077	0.1370	0.1492	0.3889	0.2499
Mutation Based	0.1558	0.0913	0.1492	0.2222	0.1673
Data Mining Based	0.0890	0.0685	0.0373	0.0000	0.0495

Table 3.10: Normalized Matrix for Alternatives-Reliability Comparison and Local Priority

Reliability	SFL Based	Program Slicing Based	Mutation Based	Data Mining Based	Local Priority
SFL Based	0.6312	0.7692	0.6383	0.3889	0.5635
Program Slicing Based	0.1842	0.1282	0.2128	0.3889	0.2384
Mutation Based	0.1384	0.1064	0.1613	0.2778	0.1923
Data Mining Based	0.0890	0.0685	0.0373	0.0000	0.0426

Table 3.11: Normalized Matrix for Alternatives-Performance Comparison and Local Priority

Performance	SFL Based	Program Slicing Based	Mutation Based	Data Mining Based	Local Priority
SFL Based	0.6841	0.7463	0.6383	0.6049	0.6541
Program Slicing Based	0.1296	0.1492	0.2128	0.2778	0.2027
Mutation Based	0.0926	0.0986	0.0555	0.0243	0.0643
Data Mining Based	0.0910	0.0746	0.0476	0.0972	0.0834

Alternatives Composite Priority and the Choice

The results from tables 3.6, 3.7, and 3.8 are then combined with Table 3.5 relative priority vector to create a new matrix:

$$\begin{pmatrix} 0.5615 & 0.5635 & 0.6541 \\ 0.2499 & 0.2384 & 0.2027 \\ 0.1673 & 0.1923 & 0.0643 \\ 0.0495 & 0.0426 & 0.0834 \end{pmatrix} \times \begin{pmatrix} 0.5823 \\ 0.3296 \\ 0.0881 \end{pmatrix} = \begin{pmatrix} 0.5703 \\ 0.2420 \\ 0.1665 \\ 0.0502 \end{pmatrix}$$

Figure 3.5: Multiplication of priority matrix by the criteria priority vector

These values may then be used to compute the composite priority vector by multiplying weighs each alternative against the priority vector. This vector will, based on the criteria, indicate the significance of the alternative.

Table 3.12: Criteria/Alternatives Classification Matrix and Composite Priority

	Accuracy	Reliability	Performance	Composite Priority
SFL Based	0.5615	0.5635	0.6541	0.5703
Program Slicing Based	0.2499	0.2384	0.2027	0.2420
Mutation Based	0.1673	0.1923	0.0643	0.1665
Data Mining Based	0.0495	0.0426	0.0834	0.0502

3.3 Conlusion

According to the results obtained using the AHP method, the global priority vector presented in Table 3.12 indicates that **Spectrum-Based Fault Localization** techniques are the most preferred option. These are followed in descending order of preference by Program Slicing-Based, Mutation-Based Fault Localization and Mutation-Based Fault Localization options.

Chapter 4

Design and Implementation

According the results of Section 2.9, SBFL techniques have the best performance among the other fault localization techniques. Additionally, the findings reveal that currently there is no open-source tool that supports the latest versions of C#, integrates with IDEs, and utilizes SBFL. This highlights the need for a new tool for automatic fault localization that fills this gap for C# developers.

Building on these findings, this dissertation proposes the creation of a new tool, *FaultDetector.NET* that supports the latest versions of C# and implements state-of-the-art SBFL techniques.

The result is two versions: the *FaultDetector.NET - Add-In*, which integrates directly into the *Visual Studio 2022* environment, and an alternative approach, *FaultDetector.NET - CLI*, which provides Command Line Interface (CLI) functionality for other usage scenarios, such as Continuous Integration and Continuous Deployment (CI/CD) integration.

4.1 FaultDetector.NET

The *FaultDetector.NET* project is openly available on the GitHub[1] as an open-source project, it allows any developer to reach it, contribute to its improvement, adapt the tool for their needs, or take this engine directly into their projects.

In terms of architecture, the project is a .Net solution with three C# projects, named *Core*, *Extension* and *Tool* as illustrated in the Figure: 4.1

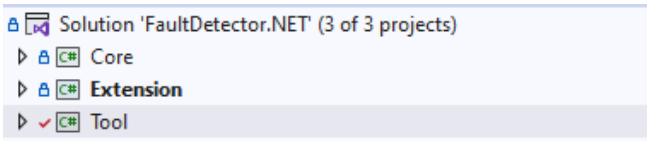


Figure 4.1: FaultDetector.NET - Solution Explorer

The *Core* is the module responsible to process the core logic of the system, which includes activities like obtain data on code coverage and calculate the suspicion metrics.

The *Extension* module is a Visual Studio 2022 add-in that will utilize the services provided by Core to offer integrated functionalities in the development environment.

[1] `https://github.com/mickaelseban/FaultDetector.NET`

The *Tool* module is CLI that also consumes the services provided by the Core module. It allows running these functionalities of the tool directly from the command line and, hence, automation and integration into the continuous integration/continuous deployment pipelines is easier.

The Figure 4.2, illustrates the dependencies between the modules mentioned.

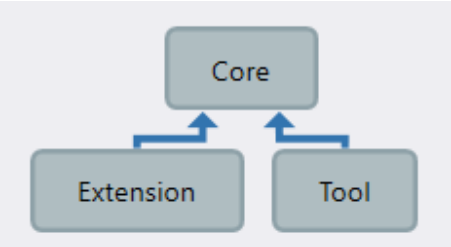


Figure 4.2: Fault Detector .NET modules dependencies

4.1.1 Fault Detector .NET - Core

4.1.2 Process

The central module of this application, where all the business logic and main processes are being executed, is the *Fault Detector .NET - Core*. *SpectrumBasedFaultLocalizationRunner* is the API of the *Fault Detector .NET - Core* that is being consumed by *Fault Detector .NET - Visual Studio Add-In* 4.1.4 and *FaultDetector.NET - CLI* 4.2. It is the centric class responsible for executing and orchestrating the whole fault location process. (See source-code in Appendix A) The process follows a sequential pipeline with a total of nine steps, as shown in Figure 4.3.

Figure 4.3: *SpectrumBasedFaultLocalizationRunner* Pipeline

Each step is detailed below:

Start of processing: When the processing is started, the *SpectrumBasedFaultLocalization-RunnerStatus* event is fired with the Running status. Any component that listens for this event will, therefore, be notified that the processing is underway.

Test Suite Execution: In order to begin the actual analysis, the tool first requires information regarding which tests are available in the test project. For this, the command `dotnet test -list-tests` is executed in an isolated process and in this way it is possible to list the tests and retrieve the complete namespace for each one.

Execution of Individual Test Cases: After that, using the previous test as the basis, it runs each of the tests individually. For every test, the *coverlet.console* tool is executed as an external process. It collects data on which lines of code were executed during each test execution and the status of the test (success/failure). This library uses instrumentation in its implementation.

Record Test Outcomes and Create Hit Spectra: In this step, both the result of each test (whether it passed or failed) is saved, and at the same time, the creation of the hit spectra occurs. The hit spectra map which parts of the code were covered by each test, and will be store on its own to a file on disc in JavaScript Object Notation (JSON) format.

Read Test Outcomes and Hit Spectra (Coverage Data): After this step, the JSON files mentioned mentioned in the previous step will be read and combined. This this information will be used in the further steps.

Construct Test Coverage Matrix: This information is then used to generate a coverage matrix. This matrix is fundamental in the analysis of SBFL, as it allows establishing a correlation between those tests that have passed or failed with those lines of code they had covered (Hit Spectra).

Calculate Suspiciousness Ranking: Suspiciousness metrics are calculated using SBFL formulas to determine which parts of the code are most likely to be faulty based on failed and successful tests.

Generate Fault Localization Report: Finally, the tool creates detailed reports that highlight the most suspicious code fragments, with the goal of making the process of identifying faults easier for the developers. When all the steps of the process have been completed, *SpectrumBasedFaultLocalizationRunnerStatus* Changed is raised again with status Finished. It reflects the situation whereby the process as a whole has been processed.

Handling errors or cancellations: In the course of processing, it may happen that the process is cancelled by the client, for instance via an *OperationCanceledException* or a critical error during the execution. In any of these cases, the method catches such exception and the terminal status of the process is signaled so that resources allocated are released properly and listeners of the interruption are notified.

4.1.3 Supported Techniques and Coefficients

Currently supports several SBFL techniques mentioned with Chapter 2, namely Tarantula C.1; Ochiai C.5; Jaccard C.3; DStar C.2; Kulczynski C.4; Rogers-Tanimoto C.6; Seban Normalization D; Tarantula & Sampaio B.4; Simple Matching & Sampaio B.2; Rogers-Tanimoto & Sampaio B.1; and Sokal Sneath & Sampaio B.3;

As discussed in the Section 2.5.1, it was implemented a dynamic calculator of CS proposed by Dr. Alberto Sampaio in (A. Sampaio, I. B. Sampaio, and Alves 2007). This dynamic CS will used by the following techniques: Tarantula & Sampaio; Simple Matching & Sampaio; Rogers-Tanimoto & Sampaio and Sokal Sneath & Sampaio .

As illustrated in the source code 4.1, NCF represents the number of failures not covered, and NCS the number of successes covered. The formula calculates the ratio between NCS and the sum of NCS and NCF, it results in an adjusted similarity measure. When there are no failures or successes ($NCF + NCS == 0$), the coefficient returns 1, assuming total symmetry.

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public static class AlbertoSampaioSymmetryCoefficient
    {
        public static double Calculate(int NCF, int NCS)
        {
            if (NCF + NCS == 0)
            {
                return 1.0;
            }

            return (double)NCS / (NCS + NCF);
        }
    }
}
```

Listing 4.1: Alberto Sampaio CS - Source Code

4.1.4 Fault Detector .NET - Visual Studio Add-In

Fault Detector .NET, can be seamlessly integrated into the *Microsoft Visual Studio 2022 IDE*. The objective is to provide a visual tool that enhances developer productivity and encourages continuous use of the *Fault Detector .NET - Add-in*.

This add-in was developed using Microsoft technologies, specifically C# and Windows Presentation Foundation (WPF). WPF enables the creation of modern, visually appealing user interfaces (GUI) for Windows applications, with the visual design coded using eXtensible Application Markup Language (XAML).

FaultDetector.NET - Add-in is available on the Visual Studio Marketplace[2], where developers can download and install it directly into their development environment, as illustrated in Figure 4.4.

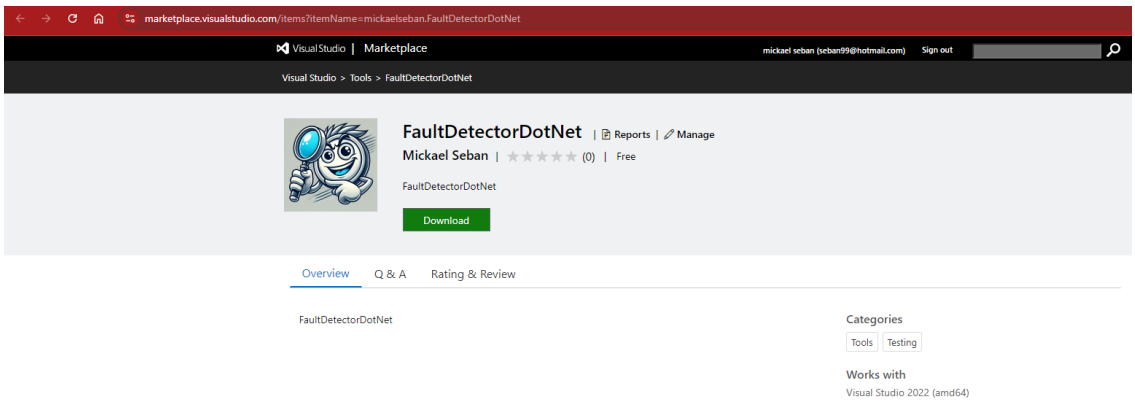


Figure 4.4: Visual Studio Marketplace - Fault Detector .NET Add-In

FaultDetector.NET - Add-in can also be installed through the Visual Studio 2022 IDE by accessing the "Manage Extensions" menu, as shown in Figure 4.5. This ensures developers can keep their tools up-to-date and remove them if necessary.

[2] https://marketplace.visualstudio.com/items?itemName=mickaelseban.FaultDetectorDotNet

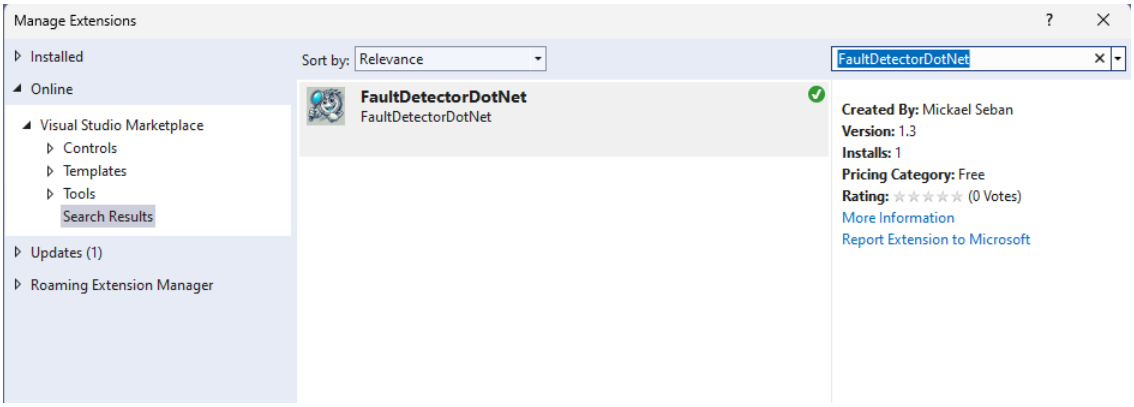


Figure 4.5: Visual Studio 2022 - Manage Extensions

The Figure 4.6 illustrates the use case of installation via Manage Extensions panel.

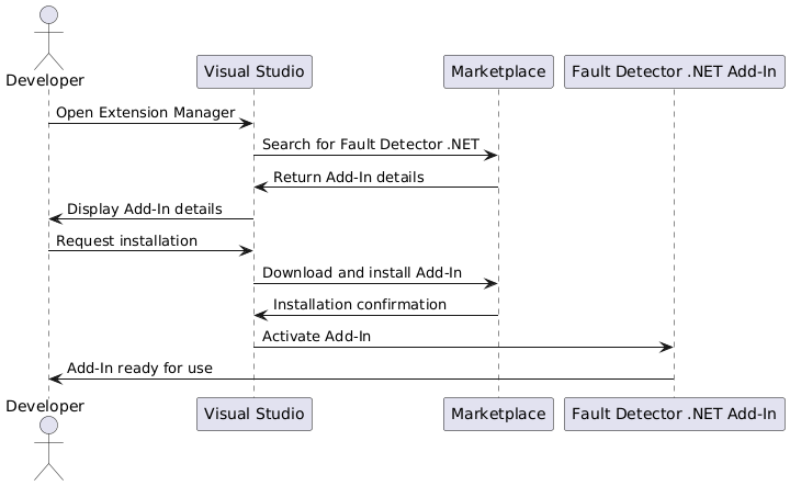


Figure 4.6: Installation Fault Detector Add-in via Manage Extensions

To use *FaultDetector.NET*, developers can easily access it through the Visual Studio 2022 menu, as shown in Figure 4.7. This menu option provides direct access to the *FaultDetector.NET* panel, where various analyses can be performed, such as fault suspicion metrics among others.

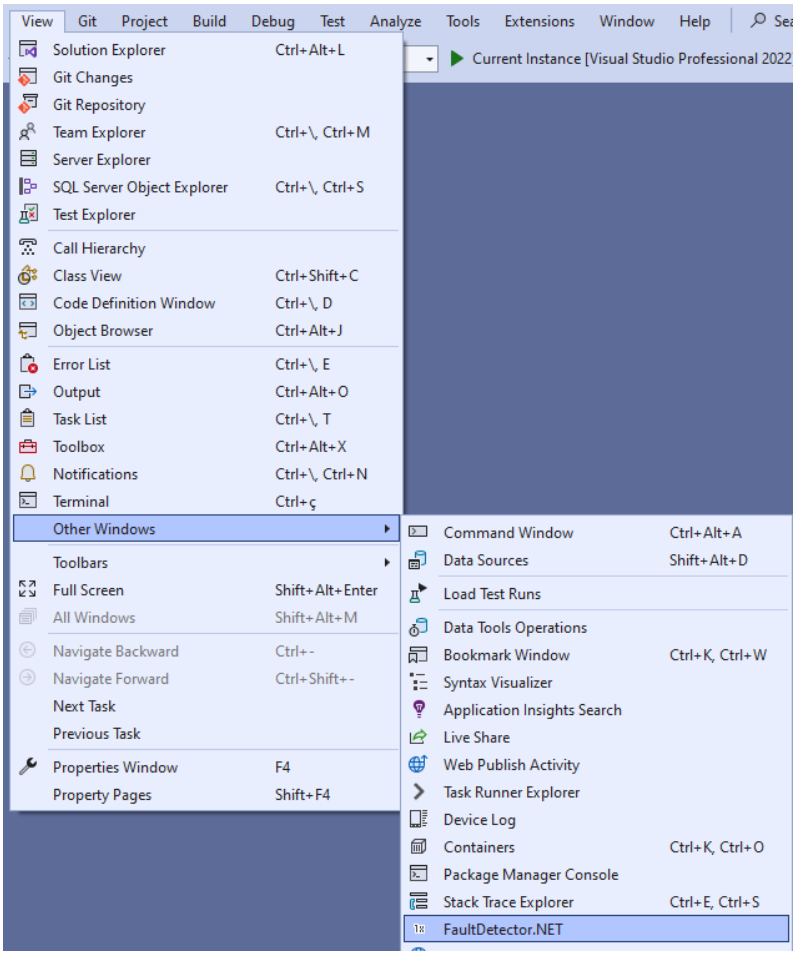


Figure 4.7: Visual Studio 2022 - Menu Show Fault Detector .NET Panel

The Figure 4.8 illustrates the use case of show the Fault Detector Panel.

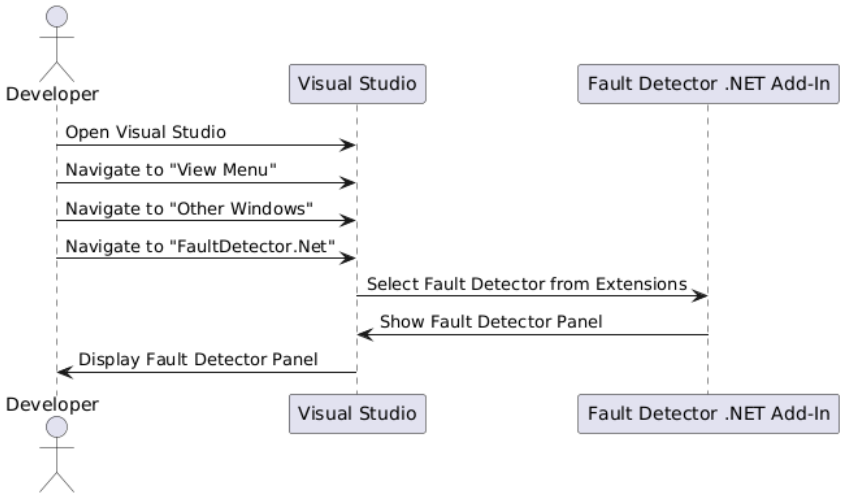


Figure 4.8: Show Fault Detector Panel

At the top of the *FaultDetector.NET* panel, there is a checkbox for each SBFL technique

available, including Tarantula; Ochiai; Jaccard; DStar; Kulczynski; Rogers-Tanimoto; Seban Normalization; Tarantula & Sampaio; Sokal Sneath & Sampaio; Simple Matching & Sampaio; and Rogers-Tanimoto & Sampaio. Below this, there is a combobox that lists all the test projects available in the solution, as illustrated in Figure 4.9.

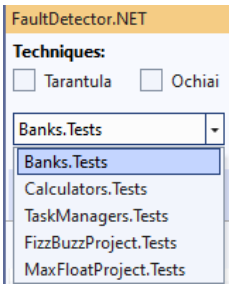


Figure 4.9: Fault Detector .NET - Tests Project Selection

When one or more of the Checkboxes corresponding to the techniques are checked, the "Run" button is enabled. This button launches the scan. Once the scan has finished, the Suspiciousousness panel in FaultDetector.NET shows the results of the analysis: the list of lines of code that have been marked according to their suspiciousness score. Figure 4.10gives an overview of the results, while Figure 4.11 depicts how all the results appear once the option "Show all Results is selected including lines with a score of 0.

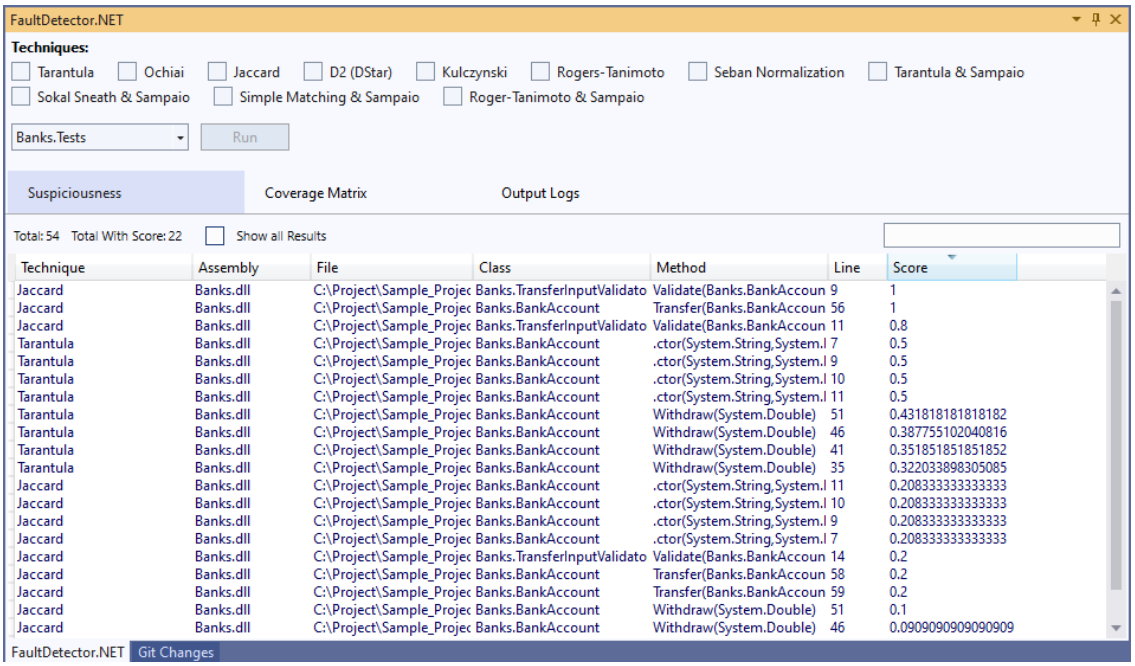


Figure 4.10: Fault Detector .NET - Suspiciousness - Results

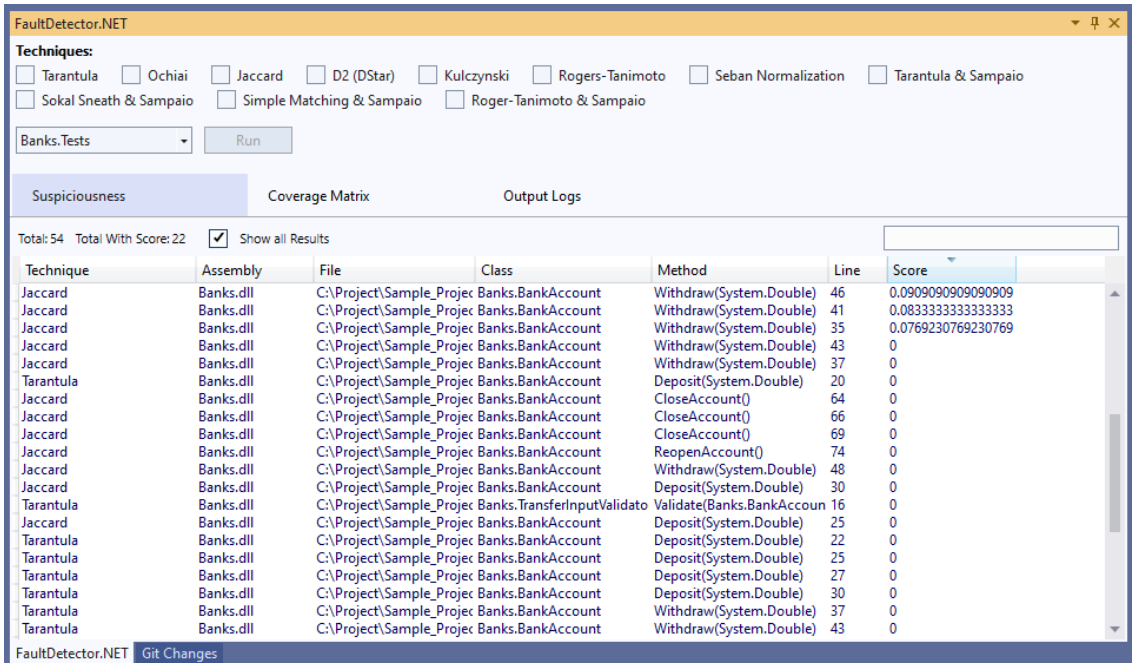


Figure 4.11: Fault Detector .NET - Suspiciousness - Show all Results

On the right of the panel, a SearchBox is provided to filter the results. It is possible to filter on any text contained in the results table of the Suspiciousness panel. Figures 4.12 and 4.13 illustrate how filters by technique and method can be set, respectively. Columns of the Suspiciousness panel support sorting in ascending/descending by sorting ; If a row in the Suspiciousness panel's table is double-clicked, the file containing the corresponding source code line is automatically opened in the IDE, positioning the cursor on the particular line. All of the functionality described above aims to enhance the usability of *FaultDetector.NET*, therefore increasing user productivity.

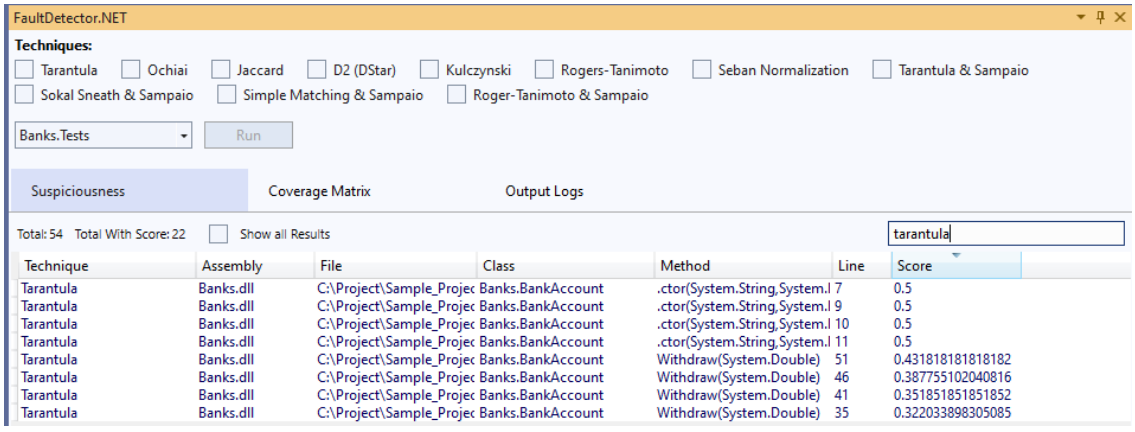


Figure 4.12: Fault Detector .NET - Suspiciousness - Filter by Technique

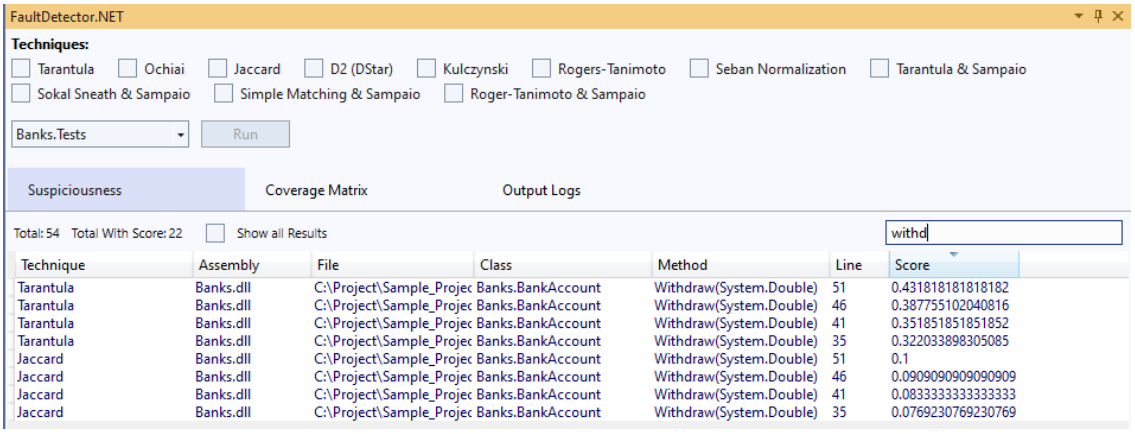


Figure 4.13: Fault Detector .NET - Suspiciousness - Filter by Method

The Coverage Matrix panel is a feature that, as the name suggests, lists in the number of times a given line of code is exercised by a given test case. (See Figure 4.14).

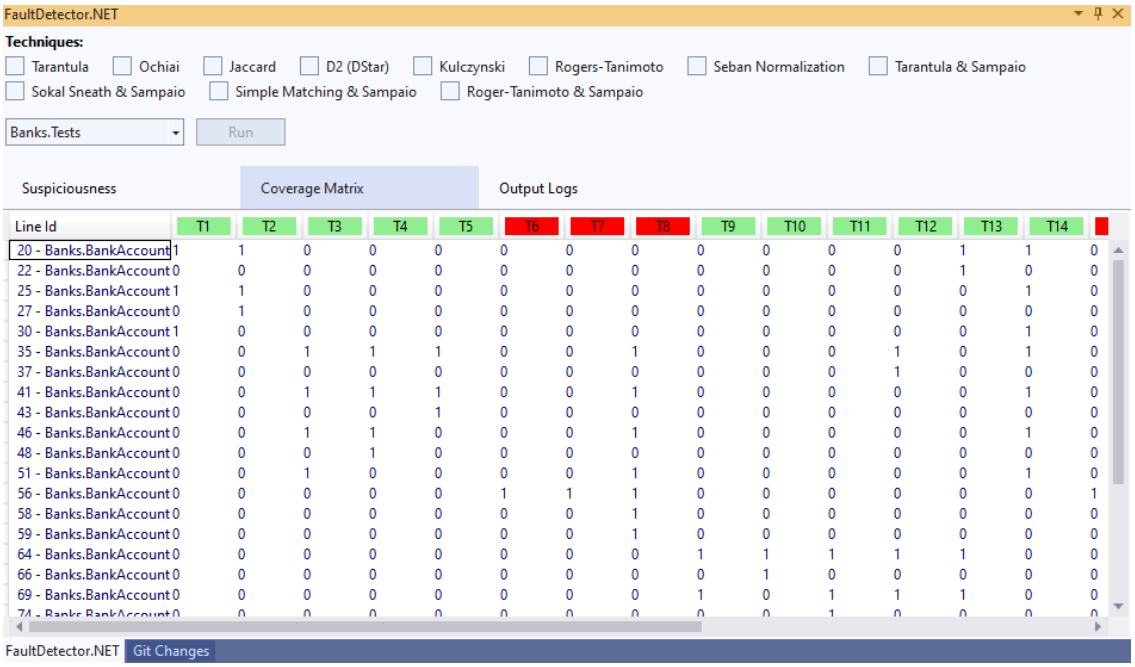


Figure 4.14: Fault Detector .NET - Coverage Matrix

The Line Id column is made up of the combination of the line number and the class namespace, representing the unique identifier. The remaining columns on the right represent the executed within the scope of the scan. The color of header indicates the test's execution status, green in the case of a pass and red in the case of a fail. Tests are identified by the letter T followed by a sequential number. To make it easier to identify the test case, when the mouse hover event occurs over the test case Id header, a Tootip appears with the name of the test case, as illustrated in 4.15.

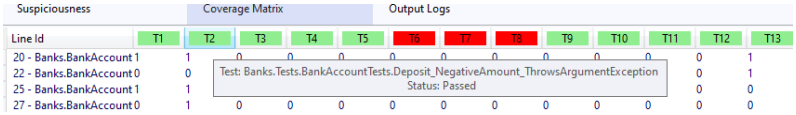


Figure 4.15: Fault Detector .NET - Coverage Matrix - Tooltip with Test Case Name

The Output Logs panel provides detailed information about the processes and results generated by *FaultDetector.NET*. The logs are written in real time when the execution of the scan. This feature is particularly useful for debugging and understanding the internal process of the tool and test execution (See Figure 4.16). The "Clear Log button, positioned on the right, deletes the logs from the panel.

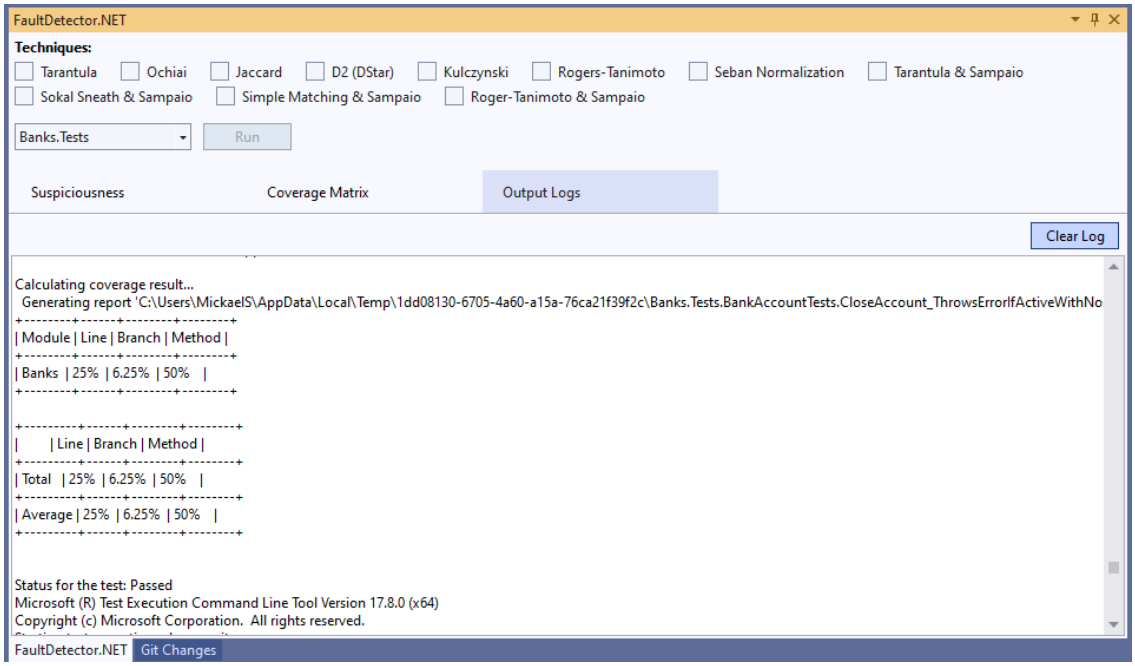


Figure 4.16: Fault Detector .NET - Output Log 2

The Figure 4.17 illustrates the use case of the process of run a scan and visualize the suspicion metrics, coverage matrix and execution logs:

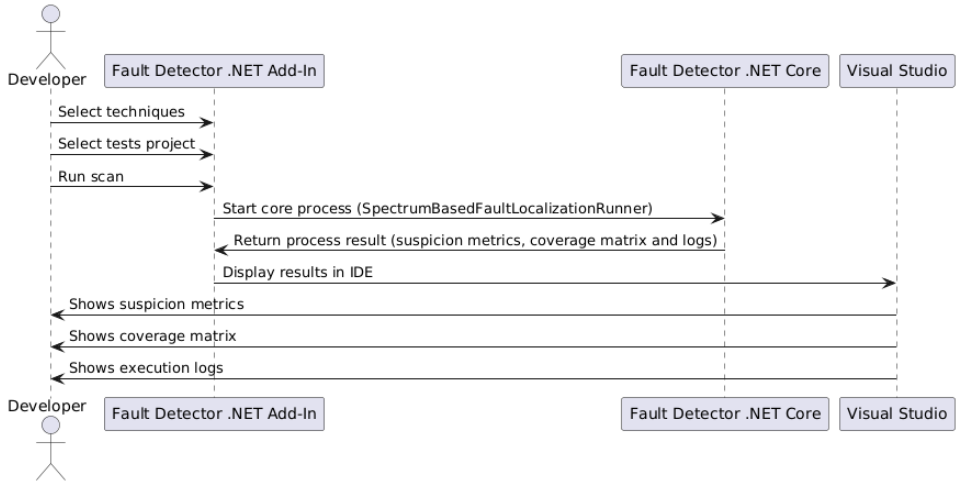


Figure 4.17: Fault Detector Add-in Scan Diagram

4.2 FaultDetector.NET - CLI

FaultDetectorDotNet CLI, like *Fault Detector .NET - Visual Studio Add-In* 4.1.4, is a tool for detecting faults in .NET projects. Both share the same functionalities. FaultDetectorDotNet CLI, is a .NET tool (dotnet tool), which means that it can be easily installed and run in any environment that supports the .NET CLI. One of the great advantages of this tool is its ability to be integrated into CI/CD (Continuous Integration/Continuous Delivery) pipelines via CLI commands, thus allowing automation in the analysis of faults throughout the development cycle so that possible problems can be identified in the build process or before a deployment.

To install the *FaultDetectorDotNet CLI* tool, you need to use the command `dotnet tool install -global FaultDetectorDotNet -version x.x.x`, as shown in Figure 4.18. This command downloads and installs the specified version of the tool globally on the system, making it accessible from any directory.

Figure 4.18: Nuget - Fault Detector .NET

The Figure 4.19 illustrates the use case of installation tool.

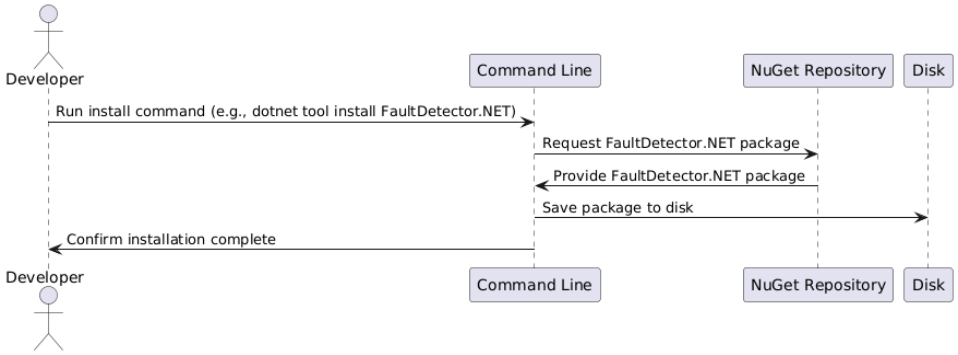


Figure 4.19: Install Fault Detector .NET CLI

After installation, we can list the tools on the system using the command: `dotnet list -global`. If successful, o *FaultDetectorDotNet* will be present, as illustrated in Figure 4.20.

```
PS C:\Project> dotnet tool list -g
Package Id                                Version       Commands
-------------------------------------------------------------------------
coverlet.console                          6.0.2         coverlet
dotnet-coverage                           17.11.0       dotnet-coverage
dotnet-reportgenerator-globaltool         5.2.5         reportgenerator
dotnet-stryker                            4.0.6         dotnet-stryker
faultdetectordotnet                       1.0.3         faultdetector
jetbrains.resharper.globaltools           2022.3.0      jb
PS C:\Project>
```

Figure 4.20: CLI - Dotnet Tool List - After Installation

If a previous version of the tool is already installed, it is possible to update it to the last version, using the command `dotnet tool update -global FaultDetectorDotNet`. Figure 4.21 shows the update process, where the tool is updated from version 1.0.1 to 1.0.2

```
PS C:\Project> dotnet tool update --global FaultDetectorDotNet --version 1.0.3
Skipping NuGet package signature verification.
Tool 'faultdetectordotnet' was successfully updated from version '1.0.2' to version '1.0.3'.
```

Figure 4.21: Update Fault Detector .NET CLI

The Figure 4.26 illustrates the use case of update the tool.

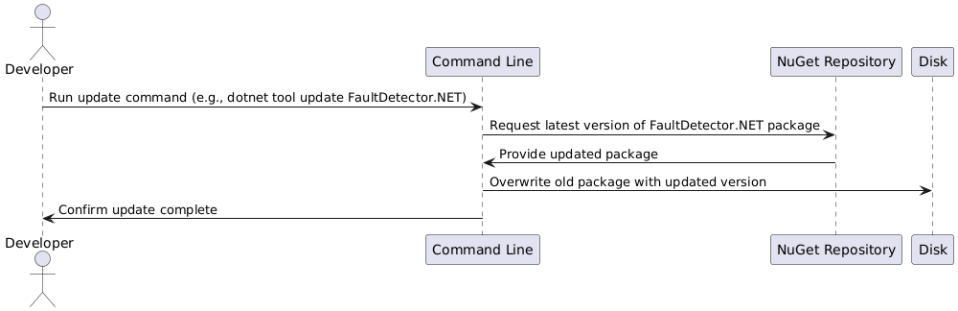


Figure 4.22: Update Fault Detector .NET CLI

Once installed, the tool can be invoked directly from the terminal using the command `faultdetector`. To see the options available, you can use the command `faultdetector`

`-help`, which displays a detailed list of arguments and options, as shown in Figure 4.23. Options include specifying the path of the test project, specifying the fault localization techniques to be applied, and the verbosity level of the logs, and many more.

```
PS C:\Project> faultdetector --help
Description:

Usage:
  FaultDetectorDotNet.Tool <test-project-path> [options]

Arguments:
  <test-project-path>  The test project full path

Options:
  -e, --exportPath <exportPath>        The path to export the results
  -v, --verbose                        Print Verbose Logs [default: False]
  -d, --debug                          Enable debug mode [default: False]
  -t, --techniques <techniques>        Fault Localization Techniques: (1 - Tarantula | 2 - Ochiai | 3 - Jaccard | 4 - D2 (DStar) | 5 - Kulczynski | 6 - Rogers-Tanimoto | 7 - Seban Normalization | 8 -
                                       Tarantula & Sampaio | 9 - Sokal Sneath & Sampaio | 10 - Roger-Tanimoto & Sampaio | 11 - Simple Matching & Sampaio) [default: Tarantula]
  -ssr, --slip-suspiciousness-results  Enable Split Suspiciousness Results. Default - Aggregated Results [default: False]
  --version                            Show version information
  -?, -h, --help                       Show help and usage information
```

Figure 4.23: Fault Detector .NET CLI - Help

After running the tool on a test project, it generates a list of lines of code classified based on different fault location techniques, such as Tarantula, Ochiai, Jaccard, among others. Figure 4.24 presents the aggregated results of these techniques, showing the most suspicious lines of code and their respective scores.

```
PS C:\Users\MickaelS> faultdetector "C:\Project\DefectsSampleProject\MaxFloatProject.Tests\MaxFloatProject.Tests.csproj" -e "C:\Project\results" -t 1,2,3,4,6
Process started
# Suspiciousness
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Technique       | Class                          | Method                                                                                | Line | Score               |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Ochiai          | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 17   | 1                   |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Jaccard         | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 17   | 1                   |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Rogers-Tanimoto | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 17   | 0.2                 |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Tarantula       | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 14   | 0.6000000000000001  |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Ochiai          | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 14   | 0.6546536707079772  |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Jaccard         | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 14   | 0.42857142857142855 |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| D2 (DStar)      | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 14   | 2.25                |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Rogers-Tanimoto | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 14   | 0.6363636363636364  |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Tarantula       | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 9    | 0.5                 |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Ochiai          | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 9    | 0.5773502691896257  |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Jaccard         | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 9    | 0.3333333333333333  |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| D2 (DStar)      | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 9    | 1.5                 |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------
| Rogers-Tanimoto | MaxFloatProject.MaxFloatProject | System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)      | 9    | 1                   |
 ---------------------------------------------------------------------------------------------------------------------------------------------------------------

Result exported as Text to C:\Project\results\9df07497-9cfc-4fb6-ba24-2f36af4489fb.txt
Exam Score exported a to C:\Project\results\9df07497-9cfc-4fb6-ba24-2f36af4489fb.json
Process completed in 20.3 seconds.
Process completed in 20.3 seconds.
```

Figure 4.24: Fault Detector .NET CLI - Aggregated Results

Is possible to split the suspiciousness results per technique as it shows the Figure 4.25. This allows a more detailed analysis of how each specific technique assesses the suspiciousness of different parts of the code.

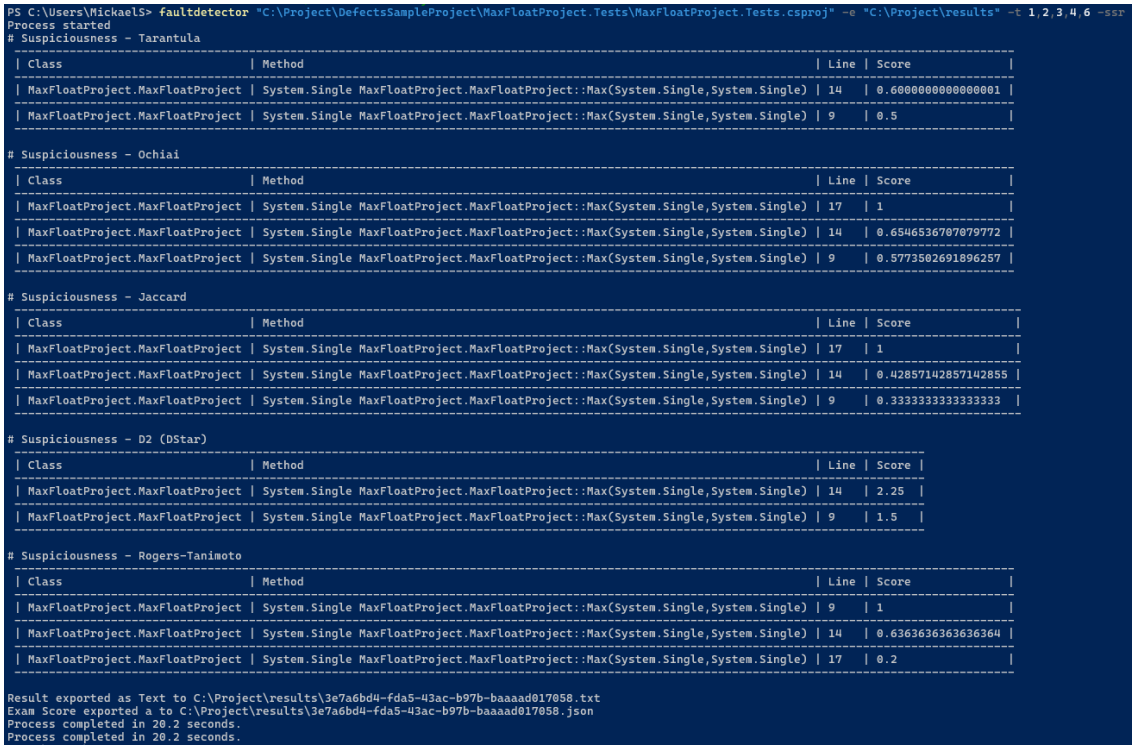


Figure 4.25: Fault Detector .NET CLI - Splited Results

The Figure 4.26 illustrates the use case of perform a scan with *Fault Detector .NET - CLI*. It involving saving the report to disk and display the results, either aggregated or splited.

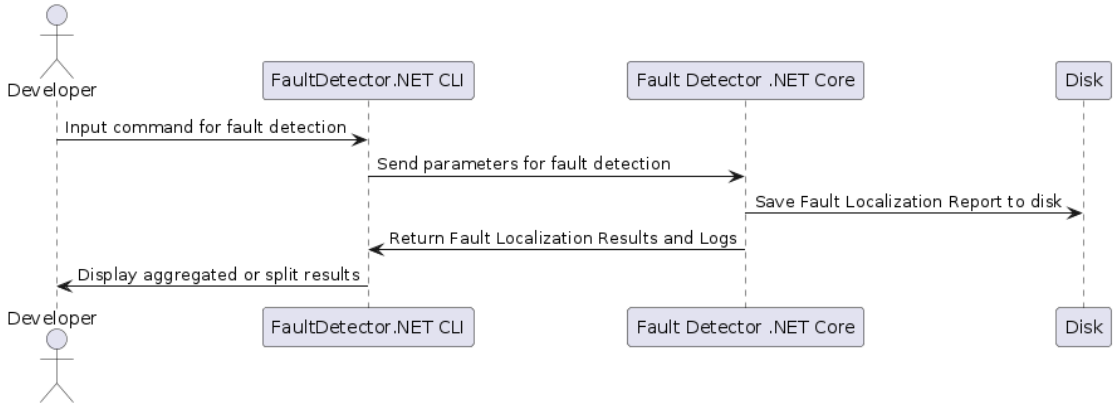


Figure 4.26: Run a Scan with Fault Detector .NET CLI

4.3 Conclusion:

A fault location solution for C# projects, called *FaultDetector.NET*, has been developed from scratch in the context of this dissertation. It supports several classic SBFL techniques, such as Tarantula, Ochiai, Jaccard and DStar, as well as alternative methods including Sampaio's Coefficient of Symmetry and Seban's Normalization. The solution is available in two forms: Through the *Visual Studio 2022* IDE extension, with a visually rich interface, and through the CLI version that facilitates its use in various development scenarios, including

Continuous Integration and Continuous Deployment pipelines. This solution has filled a gap in terms of tools that are compatible with C#, Visual Studio 2022 and available open-source, allowing anyone to contribute and extend its functionality. The detailed visualization of suspicion results and code coverage aims to help developers understand and diagnose problems in their code bases more efficiently.

Chapter 5

Evaluation and Experimentation

5.1 Evaluation

When improving any automated debugging technique, there should be some form of evaluation as to whether or not the improvements reached the proposed objectives and goals. The current chapter presents the solution evaluation and the discussion of its results. It starts by describing the methodology, presenting the metrics in which the results will be measured, detailing the investigation hypotheses, the evaluation indicators and the information sources are presented. After that, the results obtained from applying different fault localization techniques are compared, including the Sampaio Symmetry Coefficient and Seban Normalization. Finally, a summary of the different results concludes the chapter.

5.2 Methodology

Two evaluation methodologies have been used, one from the automatic debugging technique point of view and another from development. In this respect, the EXAM score metric has been used to evaluate the solution. On the other hand, from the development point of view, the automated debugging tool was submitted to a set of functional tests which aim at guaranteeing that development is controlled by needs and problem. Requirements and ensure the best quality of the final product software. The automatic software debugging tool will have to fulfill all the development good practices with the aim of ensuring reliability and maintainability, allowing easy expansion of the solution.

5.3 Metrics

The EXAM score measures the relative position of a faulty element in an ordered list of suspects that have been ranked by a fault location technique. The aim of this metric is to evaluate the performance of the debugging technique by determining how efficient it is at prioritizing faulty elements at the top of the list, making it easier to identify them.

5.4 Investigation Hypotheses

The following hypotheses can be defined in order to judge whether the objective has been fulfilled:

- **H1:** EXAM score - Lower applying Sampaio Symmetry Coefficient - Success of the improvement of the automated debugging technique applying the symmetry coefficient compared to the original fault localization techniques.
- **H2**: EXAM score - Higher applying Sampaio Symmetry Coefficient - Failure of the improved solution regarding the appliance of the symmetry coefficient to all the introduced fault localization techniques.
- **H3:** EXAM score - Lower applying Seban Normalization Technique - Success of the Seban Normalization Technique of the automated debugging applying to the others.
- **H4**: EXAM score - Higher applying Seban Normalization Technique - Failure of the Seban Normalization Technique of the automated debugging applying to the others.

5.5 Evaluation Results

In Chapter 4, refers to *Fault Localization .NET*, a solution created from scratch that provides several fault localization techniques. The real value of the solution can only be obtained when the EXAM score is compared between all of those.

To conduct a case study, a search was performed to find a dataset containing types of bugs and defects written in C#. According to *Program Repair*[1], a reference in the academic world, the available datasets only include languages like C/C++, Java, JavaScript and Python. Due to this gap, a new dataset called *Defects Sample Project* was created in C#.

5.5.1 Defects Sample Project

In order to perform a case study, the project *Defects Sample Project* [2] was developed. Is a project written C# with a suite of unit tests and a collection of reproducible bugs.

As Figure 5.1 shows, the solution of Defects Sample Project contains several projects, each representing a different domain to exemplify a real-world scenario with bugs. (See sourcecode in AppendixE)

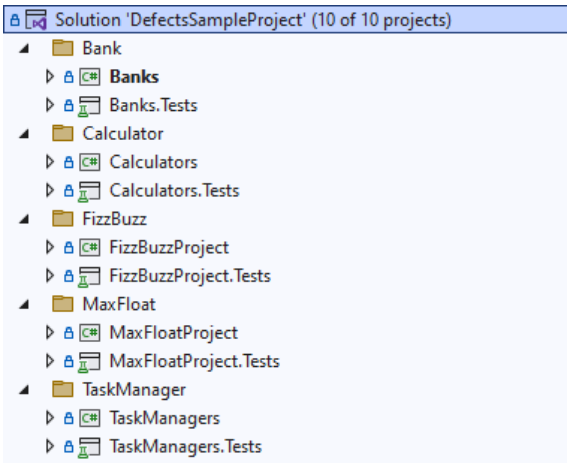


Figure 5.1: Defects Sample Project - Solution Explorer View

[1] https://program-repair.org/benchmarks.html
[2] https://github.com/mickaelseban/DefectsSampleProject

Each project with "production code" has its corresponding unit test project (see Figure 5.2).

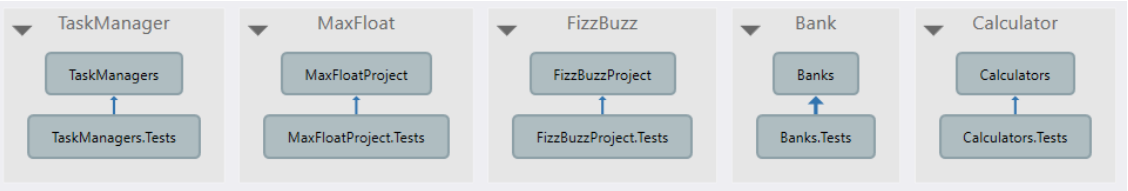


Figure 5.2: Defects Sample Project - Solution Dependencies

5.5.2 Exam score results

To evaluate the results of the techniques developed in *Fault Detector .NET - CLI*, it was conducted a comparative evaluation using the EXAM score (See chapter 2.6). To obtain the results of the EXAM score, all the projects in the *Defects Sample Project* solution were scanned through the *Fault Detector .NET - CLI*, using all the SBFL techniques provided by the tool and the results were exported as JSON. (See results: BanksF.3; CalculatorsF.3; FizzBuzzProjectF.5; MaxFloatProjectF.6; TaskManagersF.7)

The tool *EXAMScoreCalculator*[3], developed as part of this dissertation (see source code in F.1), was used to calculate the EXAM score based on the suspiciousness results generated by the *Fault Detector .NET - CLI* mentioned previously. The EXAM score was calculated and the results of this process were saved in JSON format (see results F.2).

Table 5.1: Result EXAM score

Fault Localization Technique	**EXAM score**
Roger-Tanimoto & Sampaio Symmetry Coefficient	**0.0028**
Seban Normalization	**0.0029**
Simple Matching & Sampaio Symmetry Coefficient	**0.0031**
Tarantula	**0.0032**
Jaccard	**0.0033**
Rogers-Tanimoto	**0.0033**
Sokal Sneath & Sampaio Symmetry Coefficient	**0.0033**
Ochiai	**0.0051**
Kulczynski	**0.0056**
Tarantula & Sampaio Symmetry Coefficient	**0.0071**
D2 (DStar)	**0.0250**

Based on the data provided in Table 5.1, the EXAM scores indicate the effectiveness of fault location techniques SBFL, where lower values suggest less effort is required to locate a fault. Below is an analysis of the results in the light of the proposed hypotheses.

- **H1:** EXAM score lower by applying Sampaio CS - Success of improving the automated debugging technique by applying the symmetry coefficient compared to the original fault location techniques.

  Analysis: Comparing the techniques with and without the symmetry coefficient, we observe that:

[3] https://github.com/mickaelseban/EXAMScoreCalculator

  - Tarantula: 0.0032 (original) vs. Tarantula & Sampaio: 0.0071.
  - Roger-Tanimoto: 0.0033 (original) vs. Roger-Tanimoto & Sampaio: 0.0028.

  Sampaio CS did not result in a reduction in the EXAM score for most of the techniques, except for Roger-Tanimoto & Sampaio, which showed a slight reduction. Therefore, there is partial success in H1.

- **H2:** EXAM score higher by applying Sampaio CS - Failure of the improved solution in relation to the application of the symmetry coefficient to all the fault location techniques introduced.

  Analysis: The EXAM score for some techniques increased with the symmetry coefficient, such as:

  - Tarantula: 0.0032 (original) vs. Tarantula & Sampaio: 0.0071.

  This indicates that, for some techniques such as Tarantula & Sampaio, the symmetry coefficient reduced performance, partially supporting H2.

- **H3:** EXAM score lower applying the Seban Normalization Technique - Success of the Seban Normalization Technique in improving automated debugging compared to the other techniques.

  Analysis: Seban Normalization obtained an EXAM score of 0.0029, a relatively low value, indicating good performance, compared to the other techniques. The Seban Normalization Technique was more efficient.

  Therefore, H3 is supported, as the Seban Normalization Technique was successful in reducing the fault location effort.

5.5.3 FaultDetector.NET CLI and Add-In Evaluation

In the following, the results and effectiveness of *FaultDetector.NET* will be further evaluated using metrics such as the total number of downloads from NuGet, number of installations from the Visual Studio Marketplace, and number of stars on GitHub. It is currently not possible to draw conclusions because the solution is new and the data mentioned above does not exist yet.

5.5.4 Conclusion:

Taking into account the results of the EXAM score calculated by the tool *EXAM Score Calculator* from the outcomes of the *DefectsSampleProject*, we can infer that the Roger-Tanimoto & Sampaio Symmetry Coefficient technique achieved the best performance with the lowest EXAM score (0.0028), closely followed by the Seban Normalization technique (0.0029). With regard to Symmetry Coefficient, the Roger-Tanimoto & Sampaio technique performed better than Roger-Tanimoto, while Tarantula & Sampaio had a worse EXAM score than Tarantula. Some popular techniques like Tarantula and Jaccard proved that their scores show an approximation of similar effectiveness. Both are within the range of relatively effective techniques, but they are not the most efficient techniques. In general, the difference between the scores of the techniques is quite small, except for DStar, which has a notably higher value. Regarding the adoption results of the *FaultDetector.NET*, since the tool is still in its early stages, no definitive conclusion can be drawn with regards to its wide adoption or user satisfaction. In the future, data will continuously be updated from

these sources to present a better view of the practical impact and acceptance within the community of developers.

Chapter 6

Conclusion

This chapter presents a summary of the most important results extracted from the research, showing how the proposed automatic fault localization techniques have been evaluated, and it describes the key achievements, their limitations, and possible future work.

6.1 Achieved Requirements

The overall goal throughout this research work was to present a tool and advocate for its utilization regarding fault localization within projects in C# language. The tool, *FaultDetector.NET*, was able to succinctly place together higher order spectrum-based fault localization methods along with other new methods such as Seban Normalization and Sampaio Symmetry Coefficient.

Along with the usage of these techniques, *FaultDetector.NET* represents an open-source solution of automatic fault localization, that be can integrate with *Visual Studio 2022* to extend the debugging process of C# developers more effectively. It comes in two forms: CLI form to support the CI/CD pipelines and as a Visual Studio Add-In. Moreover, the command-line version of it, meant for automating fault localization in a CI/CD process, fits well into today's modern development environments that call for continuous testing and deployment. Meanwhile, the extension for Visual Studio provides a user-friendly interface with support for multiple fault-localization techniques right inside the IDE, helping developers speed up their way of finding bugs in the course of their development process.

The techniques of fault localization support by the solution were tested with the *Defects Sample Project*, set of C# applications that contain reproducible bugs. These tests proved that *FaultDetector.NET* can handle real world applications and find defects both efficiently and effective. In addition, various techniques were implemented, such as Seban Normalization and Alberto Sampaio's asymmetry coefficient techniques. Overall, the proposed objectives were successfully achieved.

6.2 Limitations

It has not yet been possible for other developers to test the application. Only more concrete metrics, such as the number of stars on the repository, downloads from NuGet, and installations of the extension will be needed to understand adoption and the impact of *Fault Detector .NET*. Besides, the dataset (See *DefectsSampleProject* 5.5.1) on which the EXAM score was evaluated is limited in both its size and the number of bugs it contains, which restricts the generalizability of the results. The *Sampaio Symmetry Coefficient*, while useful for some

techniques such as *Roger-Tanimoto & Sampaio*, did not consistently improve performance across all fault localization methods. For instance, when applied to the *Tarantula* technique, it resulted in worse EXAM scores. This indicates that the method is not universally beneficial and its impact varies depending on the specific technique used. Another limitation involves the generation of the *hit spectra* and *test outcomes*, which are produced by *Coverlet*, a tool that relies on a heavy instrumentation process. Due to a limitation in Coverlet, tests can only be run one at a time, and it does not support parallel execution. This significantly impacts the overall performance of the tool, as the total processing time increases due to the sequential execution of tests. Furthermore, although the tool has proven effective for C# projects, it has not been thoroughly tested on larger, more complex codebases.

6.3 Future Work

There are several avenues of development that could improve the efficiency and effectiveness with which *FaultDetector.NET* carries out its work. First, it would be helpful if there was an export function whereby the user could export the results into several different formats, such as Excel, so that reporting would be easier. What's more, for better performance, the tool should not use *Coverlet*, as it relies on a pretty heavy instrumentation process, and parallel test execution is not supported. Instead, custom instrumentation at the IL level can be developed, allowing for parallel test execution and hence greatly improving the efficiency of the whole tool.

Moreover, whereas this version focuses on C#, tool functionalities can be extended in further work for other languages from the .NET ecosystem like F# and VB.NET. In such a way, the tool will be more useful and more attractive for a greater number of developers. The other area of improvement is adding support to new techniques of SBFL. Besides, generative AI techniques will be helpful in not only finding defects but also suggesting their repair and remediation, making the debugging process faster and more automated.

Addressing such areas would also enable *FaultDetector.NET* to become even more powerful and at the same time flexible for handling complex codebases, but in their turn provide additional features and support to the developer community.

Bibliography

Abreu, Rui, Peter Zoeteweij, and Arjan J C van Gemund (2008). "Diagnosis of multiple intermittent faults using model-based reasoning". In: *Proceedings of the 2008 ACM/IEEE International Conference on Automated Software Engineering*. ACM, pp. 405–414.

Abreu, Rui, Peter Zoeteweij, Rob Golsteijn, et al. (Nov. 2009). "A practical evaluation of spectrum-based fault localization". In: *Journal of Systems and Software* 82, pp. 1780–1792. doi: 10.1016/j.jss.2009.06.035.

Abreu, Rui, Pieter Zoeteweij, and Arjan J. C. van Gemund (2007). "On the Accuracy of Spectrum-based Fault Localization". In: *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*. Washington, DC, USA: IEEE Computer Society, pp. 89–98. doi: 10.1109/TAIC.PART.2007.10.

Adragna, P. (2008). "Software debugging techniques". In: *Proceedings of the Twentieth International Workshop on Principles of Diagnosis (DX09)*. Ed. by E. Frisk et al. Inverted CERN School of Computing (iCSC2006), pp. 71–86.

Agrawal, Hiralal and Joseph R. Horgan (1990). "Dynamic Program Slicing". In: *Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation*, pp. 246–256. isbn: 0897913647. doi: 10.1145/93542.93576.

Agrawal, Hiralal and JR Horgan (1993). "Dynamic slicing in the presence of unconstrained pointers". In: *Proceedings of the 1993 ACM SIGPLAN conference on Programming language design and implementation*. ACM, pp. 264–279.

Ajibode, Adekunle et al. (2022). "A Fault Localization Method Based on Metrics Combination". In: *Mathematics* 10.14, p. 2425. doi: 10.3390/math10142425.

Ball, Thomas, Mayur Naik, and Sriram K Rajamani (2004). "Localizing software faults by analyzing code coverage data". In: *Proceedings of the 19th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, pp. 1–10.

Bernasconi, Michele, Christine Choirat, and Raffaello Seri (Jan. 2009). *The Analytic Hierarchy Process and the Theory of Measurement*. Working Papers 56. University of Venice "Ca Foscari", Department of Economics. doi: 10.2307/27784145.

Campos, José et al. (2012). "GZoltar: an eclipse plug-in for testing and debugging". In: *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, pp. 378–381. doi: 10.1145/2351676.2351752.

Cellier, Peggy et al. (2011). "Mining software execution traces to guide software fault localization". In: *Proceedings of the 2011 International Conference on Mining Software Repositories*. IEEE, pp. 31–40.

Chandra, Satish, Thomas Reps, and Uday Shankar (2007). "Automated slicing for distributed programs". In: *Proceedings of the 6th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*. ACM, pp. 17–22.

Chen, Cheng and Nan Wang (2016). "UnitFL: A fault localization tool integrated with unit test". In: *2016 5th International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 136–142. doi: 10.1109/ICCSNT.2016.8070135.

Chilimbi, Trishul M and Matthias Hauswirth (2003). "Efficient path profiling". In: *Proceedings of the 2003 International Symposium on Code Generation and Optimization, 2003. CGO 2003.* IEEE, pp. 228–238.

Denmat, Thomas, Mireille Ducassé, and Olivier Ridoux (2005). "Tarantula revisited: A data mining approach to software fault localization". In: *Proceedings of the 2005 International Conference on Software Maintenance*. IEEE, pp. 343–353.

Golagha, Mojdeh et al. (2018a). "Aletheia: A Failure Diagnosis Toolchain". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pp. 13–16. doi: 10.1142/S021819400900426X.

– (2018b). "Aletheia: A Failure Diagnosis Toolchain". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. IEEE, pp. 13–16.

Golagha, Mojdeh et al. (2024). "Aletheia: A Failure Diagnosis Toolchain". In: *Proceedings of the [Conference Name]*. Technical University Of Munich. Munich, Germany.

Groce, Alex and Willem Visser (2006). "Model-based bug isolation". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 16.1, pp. 1–15.

Gupta, Rajiv, Mary Lou Soffa, and John Howard (1994). "Dynamic slicing method for efficient computation of program slices". In: *Proceedings of the 5th ACM SIGSOFT symposium on Software development environments*. ACM, pp. 97–106.

Gyimothy, Tibor, Arpad Beszedes, and Imre Forgacs (2001). "Relevant slicing". In: *Information and Software Technology* 43.3, pp. 155–164.

Harman, Mark and Robert M Hierons (2001). "Program slicing". In: *Formal methods and testing*. Springer, pp. 78–84.

Herstatt, Cornelius and Birgit Verworn (2001). "The "fuzzy front end" of innovation". In: *International Handbook on Innovation*, pp. 347–376.

Hirsch, Christian (2021). "A data mining approach to software fault localization". In: *Journal of Software: Evolution and Process* 33.5, e2345.

Hu, Jun et al. (2010). "Enhancing statistical debugging with non-parametric hypothesis tests". In: *Journal of Systems and Software* 83.4, pp. 593–604.

International Organization for Standardization (2017). *ISO/IEC/IEEE International Standard for Systems and Software Engineering – Vocabulary*. doi: 10.1109/IEEESTD.2017.8016712.

Jones, James A., Mary Jean Harrold, and John Stasko (2002). "Visualization of Test Information to Assist Fault Localization". In: *Proceedings of the 24th International Conference on Software Engineering*, pp. 467–477. doi: 10.1145/581339.581397.

Koen, Peter et al. (2001). "Providing clarity and a common language to the "fuzzy front end"". In: *Research-Technology Management* 44.2, pp. 46–55. doi: 10.1080/08956308.2001.11671418.

Koen, Peter A. and et al. (2002). "1 Fuzzy Front End: Effective Methods, Tools, and Techniques". In: *Proceedings of the Conference Name (if any)*. Check for any missing details such as conference name, location, and pages.

Liblit, Ben et al. (2005). "Scalable statistical bug isolation". In: *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*. ACM, pp. 15–26.

Liu, Charles et al. (2006). "SOBER: Statistical model-based bug localization". In: *Proceedings of the 2006 international symposium on Software testing and analysis*. ACM, pp. 286–295.

Lyle, James R and Mark Weiser (1987). "Automatic program bug location by program slicing". In: *Proceedings of the 2nd international conference on Computers and applications*. IEEE, pp. 877–882.

Machado, Pedro, José Creissac Campos, and Rui Abreu (2013). "MZoltar: automatic debugging of Android applications". In: *Proceedings of the 2013 International Workshop on Software Development Lifecycle for Mobile - DeMobile*. ACM. Saint Petersburg, Russian Federation, pp. 9–16.

Maruyama, Shinichi and S Matsuoka (2008). "Model-based fault localization for large software using dependency model". In: *Proceedings of the 2008 ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM, pp. 69–80.

Mateis, Christian, Markus Stumptner, and Franz Wotawa (2000). "Debugging complex programs with a dependency-based model". In: *Proceedings of the 2000 ACM SIGPLAN conference on Programming language design and implementation*. ACM, pp. 16–26.

Moon, Seokki, Jinhan Jung, and Doo-Hwan Bae (2014). "MUSE: Mutation-based fault localization". In: *Journal of Systems and Software* 95, pp. 139–156.

Papadakis, Mike and Yves Le Traon (2015). "Metallaxis-FL: Mutation-based fault localization". In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM, pp. 229–239.

Parnin, Chris and Alessandro Orso (2011). "Are Automated Debugging Techniques Actually Helping Programmers?" In: *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ISSTA '11. Toronto, Ontario, Canada, pp. 199–209. isbn: 9781450305624. doi: 10.1145/2001420.2001445. url: https://doi.org/10.1145/2001420.2001445.

Peffers, Ken et al. (2008). "A design science research methodology for information systems research". In: *Journal of management information systems* 24.3, pp. 45–77.

Reps, Thomas, Susan Horwitz, and Mooly Sagiv (1995). "Program slicing using dependence graphs". In: *Proceedings of the ACM SIGPLAN 1995 conference on Programming language design and implementation*. ACM, pp. 35–46.

Riboira, André and Rui Abreu (2010). "The GZoltar Project: A Graphical Debugger Interface". In: *Testing Practice and Research Techniques*. Ed. by Leonardo Bottaci and Gordon Fraser. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 215–218. isbn: 978-3-642-15585-7.

Saaty, Thomas (2008). "Decision making with the Analytic Hierarchy Process". In: *International Journal of Services Sciences* 1, pp. 83–98. doi: 10.1504/IJSSCI.2008.017590.

Sampaio, Alberto, Isabel B. Sampaio, and Gustavo Alves (2007). "Avaliação de Algumas Medidas de Dissimilaridade de Simetria Ajustável". In: *Proceedings of the Conference/-Workshop Name*. In: Conference or Workshop Name. Instituto Superior de Engenharia do Porto, Instituto Politécnico do Porto, Portugal. Porto, Portugal.

Sarhan, Mustafa et al. (2021). "CharmFL: An open-source spectrum-based fault localization tool for Python". In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, pp. 1642–1646.

Sayantini (2020). *What is Debugging and Why is it important?* Last accessed 10 January 2022. url: https://www.edureka.co/blog/what-is-debugging/.

Shchekotykhin, Kostyantyn, Thorsten Schmitz, and Dietmar Jannach (2016). "Sequential diagnosis of high cardinality faults in logic programs". In: *Journal of Artificial Intelligence Research* 57, pp. 445–482.

Silva, Alexandre et al. (2021). "FLACOCO: A Java Fault Localization Tool Using Spectrum-Based Fault Localization Techniques". In: *Proceedings of the 30th ACM Joint Meeting*

on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, pp. 1571–1575.

Srivastva, Shreya and Saru Dhir (2017). "Debugging approaches on various software processing levels". In: *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*. Vol. 2. IEEE, pp. 302–306. doi: 10.1109/ICECA.2017.8212821.

Tip, Frank (1995). "A survey of program slicing techniques". In: *Journal of programming languages* 3.3, pp. 121–189.

Wang, Nan et al. (May 2015). "FLAVS: A Fault Localization Add-in for Visual Studio". In: *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*. doi: 10.1109/COUFLESS.2015.8.

Wangnangg (2024). *UnitFL - Fast and Light Unit Test Explorer*. https://marketplace.visualstudio.com/items?itemName=Wangnangg.UnitFL.

Weiser, Mark (1979). "Program slicing". In: *Proceedings of the 5th international conference on Software engineering*, pp. 439–449.

Wen, M. et al. (2019). "Historical Spectrum based Fault Localization". In: *IEEE Transactions on Software Engineering*, pp. 1–1. doi: 10.1109/TSE.2019.2948158.

Wong, Eric, Vidroha Debroy, et al. (2012). "Radial basis function networks for fault localization". In: *IEEE Transactions on Reliability* 61.1, pp. 149–169.

Wong, Eric and Xue Qi (2016). "Machine learning-based fault localization". In: *IEEE Transactions on Software Engineering* 42.4, pp. 360–380.

Wong, W. Eric, Ran Gao, et al. (2016). "On the Effectiveness of Similarity Coefficients for SBFL When used in Conjunction with the Ochiai Metric". In: *Journal of Systems and Software* 117, pp. 450–464.

Wong, W. Eric, Ruizhi Gao, et al. (2016a). "A Survey on Software Fault Localization". In: *IEEE Transactions on Software Engineering* 42.8, pp. 707–740. doi: 10.1109/TSE.2016.2521368.

Wong, W. Eric, Ruizhi Gao, et al. (2016b). "A Survey on Software Fault Localization". In: *IEEE Transactions on Software Engineering* 42.8, pp. 707–740. doi: 10.1109/TSE.2016.2521368.

Wotawa, Franz (2002). "Model-based diagnosis of software: State of the art". In: *Artificial Intelligence Review* 17.3, pp. 131–150.

Yoo, S. et al. (Nov. 2014). "No pot of gold at the end of program spectrum rainbow: Greatest risk evaluation formula does not exist". In: *Proceedings of the 2014 ACM International Symposium on Software Testing and Analysis (ISSTA)*.

You, Qing and Zijiang Zheng (2012). "Statistical fault localization using weighted execution graphs". In: *Proceedings of the 2012 International Conference on Software Engineering*. IEEE, pp. 1–11.

Zhang, Xiangyu and Jing Xu (2011). "Relevant slicing based on program dynamic dependence graph". In: *Proceedings of the 33rd International Conference on Software Engineering*. ACM, pp. 755–764.

Zhang, Xifeng et al. (2011). "Crosstab: Statistical debugging using execution path profiling". In: *Proceedings of the 33rd International Conference on Software Engineering*. ACM, pp. 712–721.

Zhang, Zhenyu et al. (July 2008). "Debugging through Evaluation Sequences: A Controlled Experimental Study". In: *Proceedings of the International Computer Software and Applications Conference (COMPSAC)*. IEEE, pp. 128–135.

Zheng, Wei, Zheng Hu, and Jian Wang (2016). "Deep learning for software fault localization". In: *Proceedings of the 2016 ACM/IEEE International Conference on Software Engineering*. IEEE, pp. 404–414.

Zou, Dingding and et al. (2021). "An Empirical Study of Fault Localization Families and Their Combinations". In: *IEEE Transactions on Software Engineering* 47.2, pp. 332–347. doi: 10.1109/TSE.2019.2892102.

Appendix A

FaultDetector.NET - SpectrumBasedFaultLocalizationRunner Source Code

```
1  using System;
2  using System.Collections.Generic;
3  using System.Diagnostics;
4  using System.IO;
5  using System.Linq;
6  using System.Threading;
7  using System.Threading.Tasks;
8  using FaultDetectorDotNet.Core.Coverage;
9  using FaultDetectorDotNet.Core.Coverlet;
10 using FaultDetectorDotNet.Core.Helpers;
11 using FaultDetectorDotNet.Core.Logger;
12
13 namespace FaultDetectorDotNet.Core.Suspiciousness
14 {
15     public sealed class SpectrumBasedFaultLocalizationRunner
16     {
17         public event EventHandler<
       SpectrumBasedFaultLocalizationRunnerStatusType>
       SpectrumBasedFaultLocalizationRunnerStatusChanged;
18
19         public async Task Run(IProcessLogger logger, ReportManager
       reportManager, BuildProject buildProject,
       SuspiciousnessServiceParameters inputParameters, CancellationToken
       cancellationToken)
20         {
21             SpectrumBasedFaultLocalizationRunnerStatusChanged?.Invoke(
       this, SpectrumBasedFaultLocalizationRunnerStatusType.Running);
22             var testsMetadata = new Dictionary<string, TestMetadata>();
23             try
24             {
25                 logger.LogUserMessage("Process started");
26
27                 cancellationToken.ThrowIfCancellationRequested();
28
29                 if (await IsToolInstalled(logger, cancellationToken))
30                 {
31                     logger.LogDebugMessage("The code coverage tool is
       already installed");
32                 }
33                 else
```

```
34                  {
35                      logger.LogDebugMessage("Installing the code coverage tool...");
36                      await InstallCoverageTool(logger, cancellationToken);
37                  }
38
39                  cancellationToken.ThrowIfCancellationRequested();
40
41                  var testProjectDllPath = buildProject(inputParameters.TestProjectFullPath);
42
43                  if (string.IsNullOrEmpty(testProjectDllPath) || !File.Exists(testProjectDllPath))
44                  {
45                      logger.LogUserMessage($"The project {inputParameters.TestProjectFullPath} was not found");
46                      return;
47                  }
48
49                  testsMetadata = await ListTests(logger, inputParameters.TestProjectFullPath, cancellationToken);
50                  var tempPath = Path.Combine(Path.GetTempPath(), inputParameters.ExecutionId);
51                  Directory.CreateDirectory(tempPath);
52
53                  var coverletCoverageDatas = new Dictionary<string, CoverletCoverageData>();
54                  foreach (var test in testsMetadata)
55                  {
56                      cancellationToken.ThrowIfCancellationRequested();
57
58                      await RunDotnetTestWithCoverage(logger, test.Value, testProjectDllPath, tempPath,
59                          cancellationToken);
60                      cancellationToken.ThrowIfCancellationRequested();
61                      var coverletCoverageData = CoverletService.ReadReport(test.Value.TestCoverageReportPath);
62                      coverletCoverageDatas[test.Key] = coverletCoverageData;
63                  }
64
65                  cancellationToken.ThrowIfCancellationRequested();
66
67                  var coveragePerTest = CoverageTestSegmentedFactory.Create(coverletCoverageDatas);
68
69                  foreach (var test in testsMetadata)
70                  {
71                      test.Value.Coverage = coveragePerTest[test.Key];
72                  }
73
74                  var coverageAggregated = CoverageFactory.CreateAggregated(testsMetadata);
75                  var testCoverageMatrix = CoverageFactory.CreateCoverageMatrix(testsMetadata);
76                  var suspiciousnessResult = SuspiciousnessCalculator.Calculate(coverageAggregated, inputParameters);
77
```

```
78                  cancellationToken.ThrowIfCancellationRequested();
79
80                  var suspiciousnessRunnerResult = new
    SuspiciousnessRunnerResult
81                  {
82                      TestCoverageMatrix = testCoverageMatrix,
83                      SuspiciousnessResult = suspiciousnessResult,
84                  };
85
86                  reportManager.ReportAll(suspiciousnessRunnerResult);
87              }
88              catch (OperationCanceledException)
89              {
90                  logger.LogDebugMessage("The process was aborted from the
     user");
91              }
92              finally
93              {
94                  logger.Dispose();
95                  DeleteTestCoverageFiles(logger, testsMetadata);
96                  SpectrumBasedFaultLocalizationRunnerStatusChanged?.
    Invoke(this, SpectrumBasedFaultLocalizationRunnerStatusType.Finished)
    ;
97              }
98          }
99
100         private static void DeleteTestCoverageFiles(IProcessLogger
    logger, Dictionary<string, TestMetadata> testsMetadata)
101         {
102             foreach (var testMetadata in testsMetadata.Values)
103             {
104                 try
105                 {
106                     if (File.Exists(testMetadata.TestCoverageReportPath)
    )
107                     {
108                         File.Delete(testMetadata.TestCoverageReportPath)
    ;
109                         logger.LogDebugMessage($"File deleted: {
    testMetadata.TestCoverageReportPath}");
110                     }
111                 }
112                 catch (IOException ioEx)
113                 {
114                     logger.LogDebugMessage(
115                         $"Could not delete the file: {testMetadata.
    TestCoverageReportPath}. Error: {ioEx.Message}");
116                 }
117             }
118         }
119
120         private static async Task InstallCoverageTool(IProcessLogger
    logger, CancellationToken cancellationToken)
121         {
122             var processInfo = new ProcessStartInfo
123             {
124                 FileName = "dotnet",
125                 Arguments = "tool install --global coverlet.console",
126                 RedirectStandardOutput = true,
```

```
127                 RedirectStandardError = true,
128                 UseShellExecute = false,
129                 CreateNoWindow = true
130             };
131 
132             using (var process = Process.Start(processInfo))
133             {
134                 await WaitForExit(logger, process, "coverlet.console
    installation", cancellationToken);
135 
136                 if (process.ExitCode != 0)
137                 {
138                     var errors = await process.StandardError.
    ReadToEndAsync();
139                     if (!string.IsNullOrWhiteSpace(errors))
140                     {
141                         logger.LogDebugMessage(errors);
142                     }
143                 }
144             }
145         }
146 
147         private static async Task WaitForExit(IProcessLogger logger,
    Process process, string processName, CancellationToken
    cancellationToken)
148         {
149             process.WaitForExit();
150             while (!process.HasExited)
151             {
152                 await Task.Delay(100, cancellationToken);
153                 if (!cancellationToken.IsCancellationRequested)
154                 {
155                     continue;
156                 }
157 
158                 try
159                 {
160                     process.Kill();
161                 }
162                 catch (Exception ex)
163                 {
164                     logger.LogDebugMessage($"Error when trying to kill
    the process: {processName}", ex);
165                 }
166 
167                 cancellationToken.ThrowIfCancellationRequested();
168             }
169         }
170 
171         private static async Task RunDotnetTestWithCoverage(
    IProcessLogger logger, TestMetadata testMetadata,
172             string dllFilePath, string tempPath, CancellationToken
    cancellationToken)
173         {
174             var testCoveragePath = Path.Combine(tempPath, $"{
    SanitizeHelper.SanitizePath(testMetadata.TestQualifiedName)}.json");
175 
176             var testQualifiedNameSanitized = testMetadata.
    TestQualifiedName.Replace("(", "\\(").Replace(")", "\\)");
```

```
177             var startInfo = new ProcessStartInfo
178             {
179                 FileName = "coverlet",
180                 Arguments =
181                     $"{dllFilePath} --target \"dotnet\" --targetargs \"
    vstest {dllFilePath} --TestCaseFilter:\\\"DisplayName~{
    testQualifiedNameSanitized}\\\"\" --output \"{testCoveragePath}\" --
    format json",
182                 RedirectStandardOutput = true,
183                 RedirectStandardError = true,
184                 UseShellExecute = false,
185                 CreateNoWindow = true
186             };
187 
188             using (var process = Process.Start(startInfo))
189             {
190                 var output = await process.StandardOutput.ReadToEndAsync
    ();
191                 await WaitForExit(logger, process, "run coverlet",
    cancellationToken);
192                 logger.LogDebugMessage(output);
193 
194                 if (process.ExitCode != 0)
195                 {
196                     var errors = await process.StandardError.
    ReadToEndAsync();
197                     if (!string.IsNullOrWhiteSpace(errors))
198                     {
199                         logger.LogDebugMessage(errors);
200                     }
201                 }
202 
203                 var status = OutputParserHelper.ExtractStatus(output);
204                 testMetadata.Status = status;
205                 testMetadata.TestCoverageReportPath = testCoveragePath;
206                 logger.LogDebugMessage($"Status for the test: {status}")
    ;
207             }
208         }
209 
210         private static async Task<bool> IsToolInstalled(IProcessLogger
    logger, CancellationToken cancellationToken)
211         {
212             var processInfo = new ProcessStartInfo
213             {
214                 FileName = "dotnet",
215                 Arguments = "dotnet tool list -g",
216                 RedirectStandardOutput = true,
217                 RedirectStandardError = true,
218                 UseShellExecute = false,
219                 CreateNoWindow = true
220             };
221 
222             using (var process = Process.Start(processInfo))
223             {
224                 var output = await process.StandardOutput.ReadToEndAsync
    ();
225                 await WaitForExit(logger, process, "verify coverlet
    installation", cancellationToken);
```

```
226                 return output.Contains("coverlet");
227             }
228         }
229 
230         private static async Task<Dictionary<string, TestMetadata>>
    ListTests(IProcessLogger logger, string TestProjectFullPath,
    CancellationToken cancellationToken)
231         {
232             var processInfo = new ProcessStartInfo
233             {
234                 FileName = "dotnet",
235                 Arguments = $"dotnet test {TestProjectFullPath} --list-
    tests",
236                 RedirectStandardOutput = true,
237                 RedirectStandardError = true,
238                 UseShellExecute = false,
239                 CreateNoWindow = true
240             };
241 
242             using (var process = Process.Start(processInfo))
243             {
244                 var output = process.StandardOutput.ReadToEnd();
245                 await WaitForExit(logger, process, "dotnet list tests",
    cancellationToken);
246 
247                 var lines = output.Split(new[] { Environment.NewLine },
    StringSplitOptions.RemoveEmptyEntries);
248                 var testNames = lines.Where(line => line.StartsWith("
     ")).ToList();
249 
250                 var testsMetadata = testNames.ToDictionary(x => x,
    testQualifiedName => new TestMetadata
251                 {
252                     TestQualifiedName = testQualifiedName.Trim()
253                 });
254 
255                 return testsMetadata;
256             }
257         }
258     }
259 }
```

Listing A.1: SpectrumBasedFaultLocalizationRunner - Source Code

Appendix B

FaultDetector.NET - Sampaio Symetric Coefiecient & Techniques Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class RogerTanimotoAndAlbertoSampaioCSFormula :
     ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            var NCF = request.Counts.failCount;
            var NF = request.TotalFailedTests;
            var NCS = request.Counts.passCount;
            var NS = request.TotalPassedTests;

            if (NCF == 0)
            {
                return 0.0;
            }

            var CS = AlbertoSampaioSymmetryCoefficient.Calculate(NCF, NCS);

            double numerator = 2 * (NCF + NCS);
            var denominator = 2 * (NF + NS) + (1 + CS) * (NCF + NCS);

            if (denominator == 0)
            {
                return 0.0;
            }

            return numerator / denominator;
        }
    }
}
```

Listing B.1: Roger-Tanimoto & Alberto Sampaio Symmetry Coefficient Formula Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class SimpleMatchingAndAlbertoSampaioCSFormula :
     ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            var NCF = request.Counts.failCount;
            var NF = request.TotalFailedTests;
```

```
            var NCS = request.Counts.passCount;
            var NS = request.TotalPassedTests;

            if (NCF == 0)
            {
                return 0.0;
            }

            var CS = AlbertoSampaioSymmetryCoefficient.Calculate(NCF, NCS);

            double numerator = (double)(NCF + NCS);
            double denominator = NF + NS + CS * (NCF + NCS);

            if (denominator == 0)
            {
                return 0.0;
            }

            return numerator / denominator;
        }
    }
}
```

Listing B.2: Simple Matching & Alberto Sampaio CS Formula Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class SokalSneathAndAlbertoSampaioCSFormula :
     ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            var NCF = request.Counts.failCount;
            var NF = request.TotalFailedTests;
            var NCS = request.Counts.passCount;
            var NS = request.TotalPassedTests;

            if (NCF == 0)
            {
                return 0.0;
            }

            var CS = AlbertoSampaioSymmetryCoefficient.Calculate(NCF, NCS);

            double numerator = 2 * (NCF + NCS);
            double denominator = NF + 2 * (NCF + NCS) + NS + CS * (NCF + NCS);

            if (denominator == 0)
            {
                return 0.0;
            }

            return numerator / denominator;
        }
    }
}
```

Listing B.3: SokalSneath & Alberto Sampaio CS Formula Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class TarantulaAndAlbertoSampaioCsFormula : ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
```

```
7               var NCF = request.Counts.failCount;
8               var NF = request.TotalFailedTests;
9               var NCS = request.Counts.passCount;
10              var NS = request.TotalPassedTests;
11
12              if (NCF == 0)
13              {
14                  return 0.0;
15              }
16
17              var CS = AlbertoSampaioSymmetryCoefficient.Calculate(NCF, NCS);
18
19              var failRate = (double)NCF / NF;
20              var passRate = (double)NCS / NS;
21
22              var denominator = failRate + CS * passRate;
23
24              if (denominator == 0)
25              {
26                  return 0.0;
27              }
28
29              return failRate / denominator;
30          }
31      }
32  }
```

Listing B.4: Tarantula & Alberto Sampaio CS Formula Source Code

Appendix C

FaultDetector.NET - Other Techniques Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class TarantulaFormula : ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            var NCF = request.Counts.failCount;
            var NCS = request.Counts.passCount;
            var NF = request.TotalFailedTests;
            var NS = request.TotalPassedTests;

            if (NCF == 0)
            {
                return 0.0;
            }

            if (NF == 0 || NS == 0 || NCS == 0)
            {
                return 0.0;
            }

            var numerator = (double)NCF / NF;
            var denominator = numerator + (double)NCS / NS;

            if (denominator == 0)
            {
                return 0.0;
            }

            var suspiciousness = numerator / denominator;
            return suspiciousness;
        }
    }
}
```

Listing C.1: Tarantula Formula - Source Code

```
using System;

namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class DStarFormula : ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            int NCF = request.Counts.failCount;
            int NUF = request.TotalFailedTests - NCF;
```

```
            int NCS = request.Counts.passCount;

            if (NCF == 0)
            {
                return 0.0;
            }
            double denominator = NCS + NUF;

            if (denominator == 0)
            {
                return 0.0;
            }

            return Math.Pow(NCF, 2) / denominator;
        }
    }
}
```

Listing C.2: DStar Formula - Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class JaccardFormula : ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            int NCF = request.Counts.failCount;
            int NUF = request.TotalFailedTests - request.Counts.failCount;
            int NCS = request.Counts.passCount;

            if (NCF == 0)
            {
                return 0.0;
            }

            int denominator = NCF + NUF + NCS;

            if (denominator == 0)
            {
                return 0;
            }

            return (double)NCF / denominator;
        }
    }
}
```

Listing C.3: Jaccard Formula - Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class KulczynskiFormula : ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            int NCF = request.Counts.failCount;
            int NUF = request.TotalFailedTests - NCF;
            int NCS = request.Counts.passCount;

            if (NCF == 0)
            {
                return 0.0;
            }

            if (NCF + NUF == 0 || NCF + NCS == 0)
```

```
            {
                return 0.0;
            }

            double part1 = (double)NCF / (NCF + NUF);
            double part2 = (double)NCF / (NCF + NCS);

            return 0.5 * (part1 + part2);
        }
    }
}
```

Listing C.4: Kulczynski Formula - Source Code

```
using System;

namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed class OchiaiFormula : ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            int NCF = request.Counts.failCount;
            int NF = request.TotalFailedTests;
            int NCS = request.Counts.passCount;

            if (NCF == 0)
            {
                return 0.0;
            }

            if (NF == 0 || NCF + NCS == 0)
            {
                return 0;
            }

            return NCF / Math.Sqrt(NF * (NCF + NCS));
        }
    }
}
```

Listing C.5: Ochiai Formula - Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public class RogersTanimotoFormula : ISpectrumBasedFormula
    {
        public double Calculate(FormulaRequest request)
        {
            int NCF = request.Counts.failCount;
            int NCS = request.Counts.passCount;
            int NUF = request.TotalFailedTests - NCF;
            int NUP = request.TotalPassedTests - NCS;

            if (NCF == 0)
            {
                return 0.0;
            }

            double numerator = NCF + NCS;
            double denominator = NCF + NCS + 2 * (NUF + NUP);

            if (denominator == 0)
            {
                return 0.0;
```

```
23              }
24
25              return numerator / denominator;
26          }
27      }
28  }
```

Listing C.6: Rogers-Tanimoto Formula - Source Code

Appendix D

FaultDetector.NET - Seban Technique Source Code

```
using System.Collections.Generic;
using FaultDetectorDotNet.Core.Coverage;
using FaultDetectorDotNet.Core.Suspiciousness;

namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed partial class MickaelSebanTechnique : ITechnique
    {
        private readonly ISpectrumBasedFormula[] _formulas =
        {
            new TarantulaFormula(),
            new OchiaiFormula(),
            new JaccardFormula(),
            new KulczynskiFormula(),
            new RogersTanimotoFormula()
        };

        public TechniqueType Type => TechniqueType.MickaelSeban;

        public SuspiciousnessResult.Technique Calculate(
    CoverageAggregatedModel coverageAggregated)
        {
            var techniqueSuspiciousnessResults =
    ApplyTechniquesCalculation(coverageAggregated);

            var suspiciousnessCombined =
    CombineSuspiciousnessTechniquesResults(techniqueSuspiciousnessResults
    );
            var maxScore = GetMaxScoreSuspiciousnessCombined(
    suspiciousnessCombined);
            NormalizeSuspiciousness(suspiciousnessCombined, maxScore);
            FilterOutliers(suspiciousnessCombined);

            return suspiciousnessCombined;
        }

        private IReadOnlyList<SuspiciousnessResult.Technique>
    ApplyTechniquesCalculation(
            CoverageAggregatedModel coverageAggregated)
        {
            var techniqueResults = new List<SuspiciousnessResult.
    Technique>();

```

```
37              foreach (var formula in _formulas)
38              {
39                  var techniqueResult = new SuspiciousnessResult.Technique
    ();
40                  DefaultCalculatorHelper.DefaultApplyCalulation(
    coverageAggregated, formula, techniqueResult);
41                  techniqueResults.Add(techniqueResult);
42              }
43
44              return techniqueResults;
45          }
46      }
47  }
```

Listing D.1: Seban Technique - Calculate Source Code

```
1   namespace FaultDetectorDotNet.Core.Techniques
2   {
3       public sealed partial class MickaelSebanTechnique : ITechnique
4       {
5           private static SuspiciousnessResult.Technique
    CombineSuspiciousnessTechniquesResults(IReadOnlyList<
    SuspiciousnessResult.Technique> techniqueResults)
6           {
7               var result = new SuspiciousnessResult.Technique();
8
9               techniqueResults
10                  .SelectMany(tr => tr.Assemblies)
11                  .GroupBy(a => a.Key)
12                  .ForEach(assemblyGroup =>
13                  {
14                      var resultAssembly = result.Assemblies.
    GetValueOrDefault(assemblyGroup.Key, new SuspiciousnessResult.
    Assembly { Name = assemblyGroup.Key });
15                      result.Assemblies[assemblyGroup.Key] =
    resultAssembly;
16                      assemblyGroup
17                          .SelectMany(ag => ag.Value.Files).GroupBy(f => f
    .Key)
18                          .ForEach(fileGroup =>
19                          {
20                              var resultFile = resultAssembly.Files.
    GetValueOrDefault(fileGroup.Key, new SuspiciousnessResult.File {
    SourcePath = fileGroup.Key });
21                              resultAssembly.Files[fileGroup.Key] =
    resultFile;
22                              fileGroup
23                                  .SelectMany(fg => fg.Value.Classes)
24                                  .GroupBy(c => c.Key)
25                                  .ForEach(classGroup =>
26                                  {
27                                      var resultClass = resultFile.Classes
    .GetValueOrDefault(classGroup.Key, new SuspiciousnessResult.Class {
    Name = classGroup.Key });
28                                      resultFile.Classes[classGroup.Key] =
     resultClass;
29                                      classGroup
30                                          .SelectMany(cg => cg.Value.
    Methods)
31                                          .GroupBy(m => m.Key)
```

```
32                                          .ForEach(methodGroup =>
33                                          {
34                                              var resultMethod =
    resultClass.Methods.GetValueOrDefault(methodGroup.Key, new
    SuspiciousnessResult.Method { Signature = methodGroup.Key });
35                                              resultClass.Methods[
    methodGroup.Key] = resultMethod;
36                                              methodGroup
37                                                  .SelectMany(mg => mg.
    Value.Lines)
38                                                  .GroupBy(l => l.Key)
39                                                  .ForEach(lineGroup =>
40                                                  {
41                                                      var resultLine =
    resultMethod.Lines.GetValueOrDefault(lineGroup.Key, new
    SuspiciousnessResult.Line { Number = lineGroup.Key });
42                                                      resultLine.Score +=
    lineGroup.Sum(l => l.Value.Score);
43                                                      resultMethod.Lines[
    lineGroup.Key] = resultLine;
44                                                  });
45                                          });
46                                  });
47                          });
48                  });
49
50              return result;
51          }
52      }
53  }
```

Listing D.2: Seban Technique - Combine Suspiciousness Techniques Results Source Code

```
1   using System;
2   using FaultDetectorDotNet.Core.Suspiciousness;
3   using System.Linq;
4
5   namespace FaultDetectorDotNet.Core.Techniques
6   {
7       public sealed partial class MickaelSebanTechnique : ITechnique
8       {
9           private static double GetMaxScoreSuspiciousnessCombined(
    SuspiciousnessResult.Technique result)
10          {
11              return result.Assemblies.Values
12                  .SelectMany(a => a.Files
13                      .SelectMany(f => f.Value.Classes.Values
14                          .SelectMany(c => c.Methods.Values
15                              .SelectMany(m => m.Lines.Values
16                                  .Select(l => l.Score)))))
17                  .Max();
18          }
19
20          private static void NormalizeSuspiciousness(SuspiciousnessResult
    .Technique result, double maxScore)
21          {
22              foreach (var assembly in result.Assemblies.Values)
23              {
24                  foreach (var file in assembly.Files.Values)
```

```
                {
                    foreach (var clazz in file.Classes.Values)
                    {
                        foreach (var method in clazz.Methods.Values)
                        {
                            foreach (var lineNumber in method.Lines)
                            {
                                lineNumber.Value.Score = Math.Round(
    lineNumber.Value.Score / maxScore, 2);
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Listing D.3: Seban Technique - Get Max Score and Normalize Suspiciousness Source Code

```
namespace FaultDetectorDotNet.Core.Techniques
{
    public sealed partial class MickaelSebanTechnique : ITechnique
    {
        private static void RemoveLowScoreOutliers(SuspiciousnessResult.
    Technique result, double calculatedPercentil)
        {
            foreach (var lineKv
                     in from assembly in result.Assemblies.Values
                     from file in assembly.Files.Values
                     from clazz in file.Classes.Values
                     from method in clazz.Methods.Values
                     from lineKV in method.Lines
                     where lineKV.Value.Score < calculatedPercentil
                     select lineKV)
            {
                lineKv.Value.Score = 0;
            }
        }

        private static void FilterOutliers(SuspiciousnessResult.
    Technique result)
        {
            var linesScore = result.Assemblies.Values
                .SelectMany(a => a.Files
                    .SelectMany(f => f.Value.Classes.Values
                        .SelectMany(c => c.Methods.Values
                            .SelectMany(m => m.Lines.Values
                                .Select(l => l.Score)))))
                .ToArray();

            var percentileThreshold = CalculatePercentile(linesScore,
    90);
            RemoveLowScoreOutliers(result, percentileThreshold);
        }

        private static double CalculatePercentile(double[] linesScore,
    double percentil)
        {
```

```
36              Array.Sort(linesScore);
37              var k = percentil / 100.0 * (linesScore.Length - 1);
38              var f = (int)Math.Floor(k);
39              var c = (int)Math.Ceiling(k);
40              if (f == c)
41              {
42                  return linesScore[f];
43              }
44
45              return linesScore[f] + (k - f) * (linesScore[c] - linesScore
    [f]);
46          }
47      }
48  }
```

Listing D.4: Seban Technique - Filter Outliers Source Code

Appendix E

Defects Sample Project- Source Code

```
using System;

namespace Banks
{
    public class BankAccount
    {
        public BankAccount(string accountHolder, double initialBalance)
        {
            AccountHolder = accountHolder;
            Balance = initialBalance;
            IsActive = true;
        }

        public string AccountHolder { get; private set; }
        public double Balance { get; private set; }
        public bool IsActive { get; private set; }

        public void Deposit(double amount)
        {
            if (!IsActive)
            {
                throw new InvalidOperationException("Account is inactive
    ");
            }

            if (amount <= 0)
            {
                throw new ArgumentException("Amount must be positive");
            }

            Balance += amount;
        }

        public void Withdraw(double amount)
        {
            if (!IsActive)
            {
                throw new InvalidOperationException("Account is inactive
    ");
            }

            if (amount <= 0)
            {
```

```
42                 throw new ArgumentException("Amount must be positive");
43             }
44
45             if (amount > Balance)
46             {
47                 throw new InvalidOperationException("Insufficient funds"
    );
48             }
49
50             // Bug! Fix:  Balance -= amount;
51             Balance += amount;
52         }
53
54         public void Transfer(BankAccount targetAccount, double amount)
55         {
56             TransferInputValidator.Validate(targetAccount, this);
57
58             Withdraw(amount);
59             targetAccount.Deposit(amount);
60         }
61
62         public void CloseAccount()
63         {
64             if (Balance != 0)
65             {
66                 throw new InvalidOperationException("Cannot close
    account with non-zero balance");
67             }
68
69             IsActive = false;
70         }
71
72         public void ReopenAccount()
73         {
74             IsActive = true;
75         }
76     }
77 }
```

Listing E.1: BankAccount - Source Code

```
1  using System;
2
3  namespace Calculators
4  {
5      public class Calculator
6      {
7          public double Add(double a, double b)
8          {
9              // Bug! Fix: return a + b;
10             return a + b + 1;
11         }
12
13         public double Subtract(double a, double b)
14         {
15             return a - b;
16         }
17
18         public double Multiply(double a, double b)
19         {
```

```
20              return a * b;
21          }
22
23          public double Divide(double a, double b)
24          {
25              if (b == 0)
26              {
27                  // Bug! Fix: throw new DivideByZeroException("Division
    by zero is not allowed.");
28                  throw new Exception("Division by zero is not allowed.");
29              }
30
31              return a / b;
32          }
33      }
34  }
```

Listing E.2: Calculator - Source Code

```
1   namespace FizzBuzzProject
2   {
3       public class FizzBuzz
4       {
5           public string GetFizzBuzz(int number)
6           {
7               // Bug! Fix:
8               //if (number == 0)
9               //{
10              //    return "0";
11              //}
12
13              if (number % 3 == 0 && number % 5 == 0)
14              {
15                  return "FizzBuzz";
16              }
17
18              if (number % 3 == 0)
19              {
20                  return "Fizz";
21              }
22
23              if (number % 5 == 0)
24              {
25                  return "Buzz";
26              }
27
28              // Bug! Fix: return number.ToString();
29              return (number + 1).ToString();
30          }
31      }
32  }
```

Listing E.3: FizzBuzz - Source Code

```
1   using System;
2
3   namespace MaxFloatProject
4   {
5       public static class MaxFloatProject
6       {
```

```
        public static float Max(float a, float b)
        {
            if (float.IsNaN(a))
            {
                return b;
            }

            if (float.IsNaN(b))
            {
                // Bug! Fix: return a;
                return b;
            }

            return Math.Max(a, b);
        }
    }
}
```

Listing E.4: MaxFloatProject - Source Code

```
namespace TaskManagers
{
    public class Task
    {
        public Task(int id, string title, string description, string
    category, bool isCompleted, int priority)
        {
            Id = id;
            // Bug! Fix:  Title = title;
            Title = description;
            Description = description;
            IsCompleted = isCompleted;
            // Bug! Fix:  Category = category;
            Category = title;
            Priority = priority;
        }

        public int Id { get; private set; }
        public string Title { get; private set; }
        public string Description { get; private set; }
        public bool IsCompleted { get; private set; }
        public string Category { get; private set; }
        public int Priority { get; private set; }

        public void Update(string title, string description, string
    category, int priority)
        {
            Title = title;
            // Bug! Fix: Description = description;
            Description = title;
            // Bug! Fix: Category = category;
            Category = title;
            Priority = priority;
        }
    }
}
```

Listing E.5: Task - Source Code

```
using System.Collections.Generic;
using System.Linq;

namespace TaskManagers
{
    public class TaskManager
    {
        private readonly List<Task> _tasks;
        private int _nextId;

        public TaskManager()
        {
            _tasks = new List<Task>();
            _nextId = 1;
        }

        public Task AddTask(string title, string description, string category, int priority)
        {
            var task = new Task(_nextId++, title, description, category, false, priority);
            _tasks.Add(task);
            return task;
        }

        public bool RemoveTask(int id)
        {
            var task = _tasks.SingleOrDefault(t => t.Id == id);
            if (task == null)
            {
                return false;
            }

            _tasks.Remove(task);
            return true;
        }

        public Task GetTask(int id)
        {
            return _tasks.SingleOrDefault(t => t.Id == id);
        }

        public List<Task> GetAllTasks()
        {
            return _tasks;
        }

        public List<Task> GetTasksByCategory(string category)
        {
            return _tasks.Where(t => t.Category == category).ToList();
        }

        public List<Task> GetTasksByPriority(int priority)
        {
            // Bug! Fix: return _tasks.Where(t => t.Priority == priority).ToList();
            return _tasks.Where(t => t.Priority != priority).ToList();
        }

```

```
57          public bool UpdateTask(int id, string title, string description,
        string category, int priority)
58          {
59              var task = GetTask(id);
60              if (task == null)
61              {
62                  return false;
63              }
64
65              task.Update(title, description, category, priority);
66              return true;
67          }
68      }
69  }
```

Listing E.6: TaskManager - Source Code

```
1  using System;
2
3  namespace Banks
4  {
5      public static class TransferInputValidator
6      {
7          public static void Validate(BankAccount targetAccount,
       BankAccount thisAccount)
8          {
9              // Bug! Fix: targetAccount == null;
10             if (targetAccount != null)
11             {
12                 throw new ArgumentNullException(nameof(targetAccount));
13             }
14
15             if (targetAccount == thisAccount)
16             {
17                 throw new InvalidOperationException("Cannot transfer to
       the same account");
18             }
19         }
20     }
21 }
```

Listing E.7: TransferInputValidator - Source Code

Appendix F

EXAM Score

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using Newtonsoft.Json;

namespace EXAMScoreCalculator
{
    public sealed class Program
    {
        public static void Main(string[] args)
        {
            if (args.Length > 1 && string.Equals(args[1], "debug",
    StringComparison.OrdinalIgnoreCase))
            {
                Debugger.Launch();
            }

            if (args.Length == 0)
            {
                Console.WriteLine("Please provide the folder path
    containing the JSON files.");
                return;
            }

            var folderPath = Path.GetFullPath(args[0]);

            Console.WriteLine($"Normalized folder path: {folderPath}");

            if (!Directory.Exists(folderPath))
            {
                Console.WriteLine($"The provided path '{folderPath}' is
    invalid.");
                return;
            }

            var techniquesSuspiciousnessScores =
    GetTechniquesSuspiciousnessScores(folderPath);
            var examScore = CalculateExamScore(
    techniquesSuspiciousnessScores);
            foreach (var kv in examScore)
            {
                Console.WriteLine($"Technique: {kv.Key}");
                Console.WriteLine($"Exam Score: {kv.Value}");
            }
```

```
42
43          var jsonString = JsonConvert.SerializeObject(examScore,
    Formatting.Indented);
44
45          var filePath = Path.Combine(folderPath, $"{Guid.NewGuid().
    ToString()} - EXAM Score.json");
46          using (var streamWriter = new StreamWriter(filePath))
47          {
48              streamWriter.Write(jsonString);
49              streamWriter.Flush();
50          }
51      }
52
53      private static Dictionary<string, List<double>>
    GetTechniquesSuspiciousnessScores(string folderPath)
54      {
55          var aggregatedScores = new Dictionary<string, List<double
    >>();
56
57          foreach (var filePath in Directory.GetFiles(folderPath, "*.
    json"))
58          {
59              var jsonContent = File.ReadAllText(filePath);
60              var suspiciousnessResult = JsonConvert.DeserializeObject
    <SuspiciousnessResult>(jsonContent);
61
62              foreach (var techniqueEntry in suspiciousnessResult.
    Techniques)
63              {
64                  var techniqueName = techniqueEntry.Key;
65
66                  if (!aggregatedScores.ContainsKey(techniqueName))
67                  {
68                      aggregatedScores[techniqueName] = [];
69                  }
70
71                  var technique = techniqueEntry.Value;
72
73                  foreach (var line in
74                           from assemblyEntry in technique.Assemblies
75                           select assemblyEntry.Value into assembly
76                           from fileEntry in assembly.Files
77                           select fileEntry.Value into file
78                           from classEntry in file.Classes
79                           select classEntry.Value into classObj from
    methodEntry in classObj.Methods
80                           select methodEntry.Value into method from
    lineEntry in method.Lines
81                           select lineEntry.Value)
82                  {
83                      aggregatedScores[techniqueName].Add(line.Score);
84                  }
85              }
86          }
87
88          return aggregatedScores;
89      }
90
```

```
91          public static Dictionary<string, double> CalculateExamScore(
    Dictionary<string, List<double>> techniquesScores)
92          {
93              var examScores = new Dictionary<string, double>();
94
95              foreach ((var techniqueType, var scores) in techniquesScores
    )
96              {
97                  var examScore = Math.Round(scores.Sum(x => x / scores.
    Count) / scores.Count, 4);
98
99                  examScores.Add(techniqueType, examScore);
100             }
101
102             return examScores;
103         }
104     }
105 }
```

Listing F.1: EXAM Score Calculator - Source Code

```
1  {
2    "Tarantula": 0.0032,
3    "Ochiai": 0.0051,
4    "Jaccard": 0.0033,
5    "D2 (DStar)": 0.025,
6    "Kulczynski": 0.0056,
7    "Rogers-Tanimoto": 0.0033,
8    "Seban Normalization": 0.0029,
9    "Tarantula & Sampaio": 0.0071,
10   "Sokal Sneath & Sampaio": 0.0033,
11   "Roger-Tanimoto & Sampaio": 0.0028,
12   "Simple Matching & Sampaio": 0.0031
13 }
```

Listing F.2: Fault Detector .NET - Techniques EXAM Score

```
1  {
2    "Techniques": {
3      "Tarantula": {
4        "Assemblies": {
5          "Banks.dll": {
6            "Name": "Banks.dll",
7            "Files": {
8              "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
9                "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.
    cs",
10               "Classes": {
11                 "Banks.BankAccount": {
12                   "Name": "Banks.BankAccount",
13                   "Methods": {
14                     "System.String Banks.BankAccount::get_AccountHolder()": {
15                       "Signature": "System.String Banks.BankAccount::
    get_AccountHolder()",
16                       "Lines": {}
17                     },
18                     "System.Double Banks.BankAccount::get_Balance()": {
19                       "Signature": "System.Double Banks.BankAccount::get_Balance()",
20                       "Lines": {}
21                     },
22                     "System.Boolean Banks.BankAccount::get_IsActive()": {
23                       "Signature": "System.Boolean Banks.BankAccount::get_IsActive()
    ",
24                       "Lines": {}
```

```
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.
    Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.16666666666666669
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.1891891891891892
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.21875
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Withdraw(System.
    Double)",
                      "Lines": {
                        "35": {
                          "Number": 35,
                          "Score": 0.7368421052631579
                        },
                        "37": {
                          "Number": 37,
                          "Score": 0.0
                        },
                        "40": {
                          "Number": 40,
                          "Score": 0.8076923076923076
                        },
                        "42": {
                          "Number": 42,
                          "Score": 0.0
                        },
                        "45": {
                          "Number": 45,
                          "Score": 0.8936170212765958
                        },
                        "47": {
                          "Number": 47,
                          "Score": 0.0
                        },
                        "51": {
                          "Number": 51,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                      "Lines": {
                        "56": {
```

```
                        "Number": 56,
                        "Score": 0.0
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.0
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::ReopenAccount()": {
                    "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                    "Lines": {
                      "74": {
                        "Number": 74,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::.ctor(System.String,System.
    Double)": {
                    "Signature": "System.Void Banks.BankAccount::.ctor(System.
    String,System.Double)",
                    "Lines": {
                      "7": {
                        "Number": 7,
                        "Score": 0.5
                      },
                      "9": {
                        "Number": 9,
                        "Score": 0.5
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.5
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.5
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
```

```
            "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\
    TransferInputValidator.cs",
            "Classes": {
              "Banks.TransferInputValidator": {
                "Name": "Banks.TransferInputValidator",
                "Methods": {
                  "System.Void Banks.TransferInputValidator::Validate(Banks.
    BankAccount,Banks.BankAccount)": {
                    "Signature": "System.Void Banks.TransferInputValidator::
    Validate(Banks.BankAccount,Banks.BankAccount)",
                    "Lines": {
                      "10": {
                        "Number": 10,
                        "Score": 0.0
                      },
                      "12": {
                        "Number": 12,
                        "Score": 0.0
                      },
                      "15": {
                        "Number": 15,
                        "Score": 0.0
                      },
                      "17": {
                        "Number": 17,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Ochiai": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.
    cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::
    get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()
    ",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.
    Double)",
                      "Lines": {
```

```
                        "20": {
                          "Number": 20,
                          "Score": 0.11180339887498948
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.11952286093343936
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.12909944487358055
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Withdraw(System.
    Double)",
                      "Lines": {
                        "35": {
                          "Number": 35,
                          "Score": 0.6324555320336759
                        },
                        "37": {
                          "Number": 37,
                          "Score": 0.0
                        },
                        "40": {
                          "Number": 40,
                          "Score": 0.6708203932499369
                        },
                        "42": {
                          "Number": 42,
                          "Score": 0.0
                        },
                        "45": {
                          "Number": 45,
                          "Score": 0.7171371656006361
                        },
                        "47": {
                          "Number": 47,
                          "Score": 0.0
                        },
                        "51": {
                          "Number": 51,
                          "Score": 0.7745966692414834
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                      "Lines": {
                        "56": {
                          "Number": 56,
                          "Score": 0.7071067811865475
                        },
                        "58": {
                          "Number": 58,
```

```
                        "Score": 0.31622776601683794
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.31622776601683794
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::ReopenAccount()": {
                    "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                    "Lines": {
                      "74": {
                        "Number": 74,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::.ctor(System.String,System.
    Double)": {
                    "Signature": "System.Void Banks.BankAccount::.ctor(System.
    String,System.Double)",
                    "Lines": {
                      "7": {
                        "Number": 7,
                        "Score": 0.6454972243679028
                      },
                      "9": {
                        "Number": 9,
                        "Score": 0.6454972243679028
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.6454972243679028
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.6454972243679028
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\
    TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
```

```
                "Methods": {
                  "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)": {
                    "Signature": "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)",
                    "Lines": {
                      "10": {
                        "Number": 10,
                        "Score": 0.7071067811865475
                      },
                      "12": {
                        "Number": 12,
                        "Score": 0.6324555320336759
                      },
                      "15": {
                        "Number": 15,
                        "Score": 0.31622776601683794
                      },
                      "17": {
                        "Number": 17,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Jaccard": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.058823529411764705
                        },
                        "22": {
```

```
                        "Number": 22,
                        "Score": 0.0
                      },
                      "25": {
                        "Number": 25,
                        "Score": 0.0625
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.06666666666666667
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Withdraw(System.
    Double)",
                    "Lines": {
                      "35": {
                        "Number": 35,
                        "Score": 0.46153846153846156
                      },
                      "37": {
                        "Number": 37,
                        "Score": 0.0
                      },
                      "40": {
                        "Number": 40,
                        "Score": 0.5
                      },
                      "42": {
                        "Number": 42,
                        "Score": 0.0
                      },
                      "45": {
                        "Number": 45,
                        "Score": 0.5454545454545454
                      },
                      "47": {
                        "Number": 47,
                        "Score": 0.0
                      },
                      "51": {
                        "Number": 51,
                        "Score": 0.6
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                    "Lines": {
                      "56": {
                        "Number": 56,
                        "Score": 0.5
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.1
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.1
```

```
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::CloseAccount()": {
                      "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                      "Lines": {
                        "64": {
                          "Number": 64,
                          "Score": 0.0
                        },
                        "66": {
                          "Number": 66,
                          "Score": 0.0
                        },
                        "69": {
                          "Number": 69,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::ReopenAccount()": {
                      "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                      "Lines": {
                        "74": {
                          "Number": 74,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::.ctor(System.String,System.
    Double)": {
                      "Signature": "System.Void Banks.BankAccount::.ctor(System.
    String,System.Double)",
                      "Lines": {
                        "7": {
                          "Number": 7,
                          "Score": 0.4166666666666667
                        },
                        "9": {
                          "Number": 9,
                          "Score": 0.4166666666666667
                        },
                        "10": {
                          "Number": 10,
                          "Score": 0.4166666666666667
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.4166666666666667
                        }
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\
    TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
                  "Methods": {
                    "System.Void Banks.TransferInputValidator::Validate(Banks.
    BankAccount,Banks.BankAccount)": {
                      "Signature": "System.Void Banks.TransferInputValidator::
    Validate(Banks.BankAccount,Banks.BankAccount)",
```

```
                    "Lines": {
                      "10": {
                        "Number": 10,
                        "Score": 0.5
                      },
                      "12": {
                        "Number": 12,
                        "Score": 0.4
                      },
                      "15": {
                        "Number": 15,
                        "Score": 0.1
                      },
                      "17": {
                        "Number": 17,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "D2 (DStar)": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.
    cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::
    get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()
    ",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.
    Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0625
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
```

```
                        "Score": 0.06666666666666667
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.07142857142857142
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Withdraw(System.
    Double)",
                    "Lines": {
                      "35": {
                        "Number": 35,
                        "Score": 5.142857142857143
                      },
                      "37": {
                        "Number": 37,
                        "Score": 0.0
                      },
                      "40": {
                        "Number": 40,
                        "Score": 6.0
                      },
                      "42": {
                        "Number": 42,
                        "Score": 0.0
                      },
                      "45": {
                        "Number": 45,
                        "Score": 7.2
                      },
                      "47": {
                        "Number": 47,
                        "Score": 0.0
                      },
                      "51": {
                        "Number": 51,
                        "Score": 9.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                    "Lines": {
                      "56": {
                        "Number": 56,
                        "Score": 5.0
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.1111111111111111
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.1111111111111111
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
```

```
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::ReopenAccount()": {
                    "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                    "Lines": {
                      "74": {
                        "Number": 74,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::.ctor(System.String,System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::.ctor(System.String,System.Double)",
                    "Lines": {
                      "7": {
                        "Number": 7,
                        "Score": 7.142857142857143
                      },
                      "9": {
                        "Number": 9,
                        "Score": 7.142857142857143
                      },
                      "10": {
                        "Number": 10,
                        "Score": 7.142857142857143
                      },
                      "11": {
                        "Number": 11,
                        "Score": 7.142857142857143
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs",
            "Classes": {
              "Banks.TransferInputValidator": {
                "Name": "Banks.TransferInputValidator",
                "Methods": {
                  "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)": {
                    "Signature": "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)",
                    "Lines": {
                      "10": {
                        "Number": 10,
                        "Score": 5.0
                      },
```

```
                        "12": {
                          "Number": 12,
                          "Score": 2.6666666666666665
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.1111111111111111
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Kulczynski": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.
    cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::
    get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()
    ",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.
    Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.1125
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.12142857142857143
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
```

```
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.13333333333333333
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Withdraw(System.
    Double)",
                      "Lines": {
                        "35": {
                          "Number": 35,
                          "Score": 0.6333333333333333
                        },
                        "37": {
                          "Number": 37,
                          "Score": 0.0
                        },
                        "40": {
                          "Number": 40,
                          "Score": 0.675
                        },
                        "42": {
                          "Number": 42,
                          "Score": 0.0
                        },
                        "45": {
                          "Number": 45,
                          "Score": 0.7285714285714285
                        },
                        "47": {
                          "Number": 47,
                          "Score": 0.0
                        },
                        "51": {
                          "Number": 51,
                          "Score": 0.8
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                      "Lines": {
                        "56": {
                          "Number": 56,
                          "Score": 0.75
                        },
                        "58": {
                          "Number": 58,
                          "Score": 0.55
                        },
                        "59": {
                          "Number": 59,
                          "Score": 0.55
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::CloseAccount()": {
                      "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                      "Lines": {
                        "64": {
                          "Number": 64,
                          "Score": 0.0
                        },
```

```
                        "66": {
                          "Number": 66,
                          "Score": 0.0
                        },
                        "69": {
                          "Number": 69,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::ReopenAccount()": {
                      "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                      "Lines": {
                        "74": {
                          "Number": 74,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::.ctor(System.String,System.
    Double)": {
                      "Signature": "System.Void Banks.BankAccount::.ctor(System.
    String,System.Double)",
                      "Lines": {
                        "7": {
                          "Number": 7,
                          "Score": 0.7083333333333334
                        },
                        "9": {
                          "Number": 9,
                          "Score": 0.7083333333333334
                        },
                        "10": {
                          "Number": 10,
                          "Score": 0.7083333333333334
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.7083333333333334
                        }
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\
    TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
                  "Methods": {
                    "System.Void Banks.TransferInputValidator::Validate(Banks.
    BankAccount,Banks.BankAccount)": {
                      "Signature": "System.Void Banks.TransferInputValidator::
    Validate(Banks.BankAccount,Banks.BankAccount)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.75
                        },
                        "12": {
                          "Number": 12,
                          "Score": 0.7
                        },
                        "15": {
```

```
                        "Number": 15,
                        "Score": 0.55
                      },
                      "17": {
                        "Number": 17,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Rogers-Tanimoto": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.2
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.17073170731707318
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.14285714285714285
                        }
```

```
                    }
                  },
                  "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Withdraw(System.
    Double)",
                    "Lines": {
                      "35": {
                        "Number": 35,
                        "Score": 0.23076923076923078
                      },
                      "37": {
                        "Number": 37,
                        "Score": 0.0
                      },
                      "40": {
                        "Number": 40,
                        "Score": 0.2
                      },
                      "42": {
                        "Number": 42,
                        "Score": 0.0
                      },
                      "45": {
                        "Number": 45,
                        "Score": 0.17073170731707318
                      },
                      "47": {
                        "Number": 47,
                        "Score": 0.0
                      },
                      "51": {
                        "Number": 51,
                        "Score": 0.14285714285714285
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                    "Lines": {
                      "56": {
                        "Number": 56,
                        "Score": 0.11627906976744186
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.02127659574468085
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.02127659574468085
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
```

```
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::ReopenAccount()": {
                    "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                    "Lines": {
                      "74": {
                        "Number": 74,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::.ctor(System.String,System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::.ctor(System.String,System.Double)",
                    "Lines": {
                      "7": {
                        "Number": 7,
                        "Score": 1.0
                      },
                      "9": {
                        "Number": 9,
                        "Score": 1.0
                      },
                      "10": {
                        "Number": 10,
                        "Score": 1.0
                      },
                      "11": {
                        "Number": 11,
                        "Score": 1.0
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs",
            "Classes": {
              "Banks.TransferInputValidator": {
                "Name": "Banks.TransferInputValidator",
                "Methods": {
                  "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)": {
                    "Signature": "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)",
                    "Lines": {
                      "10": {
                        "Number": 10,
                        "Score": 0.11627906976744186
                      },
                      "12": {
                        "Number": 12,
                        "Score": 0.09090909090909091
                      },
                      "15": {
                        "Number": 15,
                        "Score": 0.02127659574468085
                      },
                      "17": {
                        "Number": 17,
```

```
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Seban Normalization": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.0
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Withdraw(System.Double)",
```

```
                    "Lines": {
                      "35": {
                        "Number": 35,
                        "Score": 0.0
                      },
                      "37": {
                        "Number": 37,
                        "Score": 0.0
                      },
                      "40": {
                        "Number": 40,
                        "Score": 0.0
                      },
                      "42": {
                        "Number": 42,
                        "Score": 0.0
                      },
                      "45": {
                        "Number": 45,
                        "Score": 0.0
                      },
                      "47": {
                        "Number": 47,
                        "Score": 0.0
                      },
                      "51": {
                        "Number": 51,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                    "Lines": {
                      "56": {
                        "Number": 56,
                        "Score": 0.0
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.0
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
```

```
                    "System.Void Banks.BankAccount::ReopenAccount()": {
                      "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                      "Lines": {
                        "74": {
                          "Number": 74,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::.ctor(System.String,System.
Double)": {
                      "Signature": "System.Void Banks.BankAccount::.ctor(System.
String,System.Double)",
                      "Lines": {
                        "7": {
                          "Number": 7,
                          "Score": 1.0
                        },
                        "9": {
                          "Number": 9,
                          "Score": 1.0
                        },
                        "10": {
                          "Number": 10,
                          "Score": 1.0
                        },
                        "11": {
                          "Number": 11,
                          "Score": 1.0
                        }
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\
TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
                  "Methods": {
                    "System.Void Banks.TransferInputValidator::Validate(Banks.
BankAccount,Banks.BankAccount)": {
                      "Signature": "System.Void Banks.TransferInputValidator::
Validate(Banks.BankAccount,Banks.BankAccount)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.0
                        },
                        "12": {
                          "Number": 12,
                          "Score": 0.0
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        }
                      }
                    }
                  }
```

```
                }
              }
            }
          }
        }
      }
    },
    "Tarantula & Sampaio": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.
cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::
get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()
",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.
Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.186046511627907
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.21397379912663758
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.25149700598802394
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Withdraw(System.
Double)",
                      "Lines": {
                        "35": {
                          "Number": 35,
                          "Score": 0.8936170212765958
                        },
```

```
                        "37": {
                          "Number": 37,
                          "Score": 0.0
                        },
                        "40": {
                          "Number": 40,
                          "Score": 0.9438202247191011
                        },
                        "42": {
                          "Number": 42,
                          "Score": 0.0
                        },
                        "45": {
                          "Number": 45,
                          "Score": 0.9832775919732442
                        },
                        "47": {
                          "Number": 47,
                          "Score": 0.0
                        },
                        "51": {
                          "Number": 51,
                          "Score": 1.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                      "Lines": {
                        "56": {
                          "Number": 56,
                          "Score": 1.0
                        },
                        "58": {
                          "Number": 58,
                          "Score": 1.0
                        },
                        "59": {
                          "Number": 59,
                          "Score": 1.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::CloseAccount()": {
                      "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                      "Lines": {
                        "64": {
                          "Number": 64,
                          "Score": 0.0
                        },
                        "66": {
                          "Number": 66,
                          "Score": 0.0
                        },
                        "69": {
                          "Number": 69,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::ReopenAccount()": {
                      "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                      "Lines": {
                        "74": {
                          "Number": 74,
```

```
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::.ctor(System.String,System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::.ctor(System.String,System.Double)",
                    "Lines": {
                      "7": {
                        "Number": 7,
                        "Score": 0.631578947368421
                      },
                      "9": {
                        "Number": 9,
                        "Score": 0.631578947368421
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.631578947368421
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.631578947368421
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
                  "Methods": {
                    "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)": {
                      "Signature": "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 1.0
                        },
                        "12": {
                          "Number": 12,
                          "Score": 1.0
                        },
                        "15": {
                          "Number": 15,
                          "Score": 1.0
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
```

```
      }
    },
    "Sokal Sneath & Sampaio": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.
    cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::
    get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()
    ",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.
    Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.3404255319148936
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.3181818181818182
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.2926829268292683
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Withdraw(System.
    Double)",
                      "Lines": {
                        "35": {
                          "Number": 35,
                          "Score": 0.4
                        },
                        "37": {
                          "Number": 37,
                          "Score": 0.0
                        },
                        "40": {
```

```
                        "Number": 40,
                        "Score": 0.38095238095238093
                      },
                      "42": {
                        "Number": 42,
                        "Score": 0.0
                      },
                      "45": {
                        "Number": 45,
                        "Score": 0.358974358974359
                      },
                      "47": {
                        "Number": 47,
                        "Score": 0.0
                      },
                      "51": {
                        "Number": 51,
                        "Score": 0.3333333333333333
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                    "Lines": {
                      "56": {
                        "Number": 56,
                        "Score": 0.29411764705882354
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.07692307692307693
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.07692307692307693
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::ReopenAccount()": {
                    "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                    "Lines": {
                      "74": {
                        "Number": 74,
                        "Score": 0.0
                      }
                    }
                  },
```

```
                    "System.Void Banks.BankAccount::.ctor(System.String,System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::.ctor(System.String,System.Double)",
                      "Lines": {
                        "7": {
                          "Number": 7,
                          "Score": 0.5581395348837209
                        },
                        "9": {
                          "Number": 9,
                          "Score": 0.5581395348837209
                        },
                        "10": {
                          "Number": 10,
                          "Score": 0.5581395348837209
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.5581395348837209
                        }
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
                  "Methods": {
                    "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)": {
                      "Signature": "System.Void Banks.TransferInputValidator::Validate(Banks.BankAccount,Banks.BankAccount)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.29411764705882354
                        },
                        "12": {
                          "Number": 12,
                          "Score": 0.25
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.07692307692307693
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Roger-Tanimoto & Sampaio": {
      "Assemblies": {
```

```
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs",
              "Classes": {
                "Banks.BankAccount": {
                  "Name": "Banks.BankAccount",
                  "Methods": {
                    "System.String Banks.BankAccount::get_AccountHolder()": {
                      "Signature": "System.String Banks.BankAccount::get_AccountHolder()",
                      "Lines": {}
                    },
                    "System.Double Banks.BankAccount::get_Balance()": {
                      "Signature": "System.Double Banks.BankAccount::get_Balance()",
                      "Lines": {}
                    },
                    "System.Boolean Banks.BankAccount::get_IsActive()": {
                      "Signature": "System.Boolean Banks.BankAccount::get_IsActive()",
                      "Lines": {}
                    },
                    "System.Void Banks.BankAccount::Deposit(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Deposit(System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.25396825396825395
                        },
                        "22": {
                          "Number": 22,
                          "Score": 0.0
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.22950819672131148
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.2033898305084746
                        }
                      }
                    },
                    "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                      "Signature": "System.Void Banks.BankAccount::Withdraw(System.Double)",
                      "Lines": {
                        "35": {
                          "Number": 35,
                          "Score": 0.3
                        },
                        "37": {
                          "Number": 37,
                          "Score": 0.0
                        },
                        "40": {
                          "Number": 40,
                          "Score": 0.27586206896551724
                        },
                        "42": {
```

```
                        "Number": 42,
                        "Score": 0.0
                      },
                      "45": {
                        "Number": 45,
                        "Score": 0.25
                      },
                      "47": {
                        "Number": 47,
                        "Score": 0.0
                      },
                      "51": {
                        "Number": 51,
                        "Score": 0.2222222222222222
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                    "Lines": {
                      "56": {
                        "Number": 56,
                        "Score": 0.18867924528301888
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.04081632653061224
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.04081632653061224
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::ReopenAccount()": {
                    "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                    "Lines": {
                      "74": {
                        "Number": 74,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::.ctor(System.String,System.
    Double)": {
                    "Signature": "System.Void Banks.BankAccount::.ctor(System.
    String,System.Double)",
                    "Lines": {
```

```
                        "7": {
                          "Number": 7,
                          "Score": 0.5581395348837209
                        },
                        "9": {
                          "Number": 9,
                          "Score": 0.5581395348837209
                        },
                        "10": {
                          "Number": 10,
                          "Score": 0.5581395348837209
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.5581395348837209
                        }
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\
    TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
                  "Methods": {
                    "System.Void Banks.TransferInputValidator::Validate(Banks.
    BankAccount,Banks.BankAccount)": {
                      "Signature": "System.Void Banks.TransferInputValidator::
    Validate(Banks.BankAccount,Banks.BankAccount)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.18867924528301888
                        },
                        "12": {
                          "Number": 12,
                          "Score": 0.15384615384615385
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.04081632653061224
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Simple Matching & Sampaio": {
      "Assemblies": {
        "Banks.dll": {
          "Name": "Banks.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs": {
```

```
            "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\BankAccount.cs",
            "Classes": {
              "Banks.BankAccount": {
                "Name": "Banks.BankAccount",
                "Methods": {
                  "System.String Banks.BankAccount::get_AccountHolder()": {
                    "Signature": "System.String Banks.BankAccount::get_AccountHolder()",
                    "Lines": {}
                  },
                  "System.Double Banks.BankAccount::get_Balance()": {
                    "Signature": "System.Double Banks.BankAccount::get_Balance()",
                    "Lines": {}
                  },
                  "System.Boolean Banks.BankAccount::get_IsActive()": {
                    "Signature": "System.Boolean Banks.BankAccount::get_IsActive()",
                    "Lines": {}
                  },
                  "System.Void Banks.BankAccount::Deposit(System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Deposit(System.Double)",
                    "Lines": {
                      "20": {
                        "Number": 20,
                        "Score": 0.25806451612903225
                      },
                      "22": {
                        "Number": 22,
                        "Score": 0.0
                      },
                      "25": {
                        "Number": 25,
                        "Score": 0.23333333333333334
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.20689655172413793
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Withdraw(System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Withdraw(System.Double)",
                    "Lines": {
                      "35": {
                        "Number": 35,
                        "Score": 0.3333333333333333
                      },
                      "37": {
                        "Number": 37,
                        "Score": 0.0
                      },
                      "40": {
                        "Number": 40,
                        "Score": 0.3076923076923077
                      },
                      "42": {
                        "Number": 42,
                        "Score": 0.0
                      },
                      "45": {
```

```
                        "Number": 45,
                        "Score": 0.28
                      },
                      "47": {
                        "Number": 47,
                        "Score": 0.0
                      },
                      "51": {
                        "Number": 51,
                        "Score": 0.25
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::Transfer(Banks.BankAccount,
    System.Double)": {
                    "Signature": "System.Void Banks.BankAccount::Transfer(Banks.
    BankAccount,System.Double)",
                    "Lines": {
                      "56": {
                        "Number": 56,
                        "Score": 0.20833333333333334
                      },
                      "58": {
                        "Number": 58,
                        "Score": 0.041666666666666664
                      },
                      "59": {
                        "Number": 59,
                        "Score": 0.041666666666666664
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::CloseAccount()": {
                    "Signature": "System.Void Banks.BankAccount::CloseAccount()",
                    "Lines": {
                      "64": {
                        "Number": 64,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      },
                      "69": {
                        "Number": 69,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::ReopenAccount()": {
                    "Signature": "System.Void Banks.BankAccount::ReopenAccount()",
                    "Lines": {
                      "74": {
                        "Number": 74,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void Banks.BankAccount::.ctor(System.String,System.
    Double)": {
                    "Signature": "System.Void Banks.BankAccount::.ctor(System.
    String,System.Double)",
                    "Lines": {
                      "7": {
                        "Number": 7,
                        "Score": 0.631578947368421
                      },
```

```
                        "9": {
                          "Number": 9,
                          "Score": 0.631578947368421
                        },
                        "10": {
                          "Number": 10,
                          "Score": 0.631578947368421
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.631578947368421
                        }
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\Banks\\TransferInputValidator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Banks\\
    TransferInputValidator.cs",
              "Classes": {
                "Banks.TransferInputValidator": {
                  "Name": "Banks.TransferInputValidator",
                  "Methods": {
                    "System.Void Banks.TransferInputValidator::Validate(Banks.
    BankAccount,Banks.BankAccount)": {
                      "Signature": "System.Void Banks.TransferInputValidator::
    Validate(Banks.BankAccount,Banks.BankAccount)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.20833333333333334
                        },
                        "12": {
                          "Number": 12,
                          "Score": 0.16666666666666666
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.041666666666666664
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```


Listing F.3: Defects Sample Project - Banks - Suspiciousness Results from Fault Detector .NET CLI


```
{
  "Techniques": {
    "Tarantula": {
      "Assemblies": {
        "Calculators.dll": {
```

```
      "Name": "Calculators.dll",
      "Files": {
        "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
          "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs",
          "Classes": {
            "Calculators.Calculator": {
              "Name": "Calculators.Calculator",
              "Methods": {
                "System.Double Calculators.Calculator::Add(System.Double,System.Double)": {
                  "Signature": "System.Double Calculators.Calculator::Add(System.Double,System.Double)",
                  "Lines": {
                    "10": {
                      "Number": 10,
                      "Score": 0.0
                    }
                  }
                },
                "System.Double Calculators.Calculator::Subtract(System.Double,System.Double)": {
                  "Signature": "System.Double Calculators.Calculator::Subtract(System.Double,System.Double)",
                  "Lines": {
                    "15": {
                      "Number": 15,
                      "Score": 0.0
                    }
                  }
                },
                "System.Double Calculators.Calculator::Multiply(System.Double,System.Double)": {
                  "Signature": "System.Double Calculators.Calculator::Multiply(System.Double,System.Double)",
                  "Lines": {
                    "20": {
                      "Number": 20,
                      "Score": 0.0
                    }
                  }
                },
                "System.Double Calculators.Calculator::Divide(System.Double,System.Double)": {
                  "Signature": "System.Double Calculators.Calculator::Divide(System.Double,System.Double)",
                  "Lines": {
                    "25": {
                      "Number": 25,
                      "Score": 0.5
                    },
                    "28": {
                      "Number": 28,
                      "Score": 0.0
                    },
                    "31": {
                      "Number": 31,
                      "Score": 0.0
                    }
                  }
                }
              }
            }
          }
        }
      }
```

```
      }
    },
    "Ochiai": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
.Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.8164965809277261
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(
System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Multiply(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(
System.Double,System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Divide(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Divide(
System.Double,System.Double)",
                      "Lines": {
                        "25": {
                          "Number": 25,
                          "Score": 0.4082482904638631
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.5773502691896258
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.0
                        }
                      }
                    }
                  }
```

```
                }
              }
            }
          }
        }
      }
    },
    "Jaccard": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
.Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.6666666666666666
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(
System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Multiply(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(
System.Double,System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Divide(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Divide(
System.Double,System.Double)",
                      "Lines": {
                        "25": {
                          "Number": 25,
                          "Score": 0.25
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.3333333333333333
                        },
                        "31": {
                          "Number": 31,
```

```
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "D2 (DStar)": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
.Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 4.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(
System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Multiply(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(
System.Double,System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Divide(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Divide(
System.Double,System.Double)",
                      "Lines": {
                        "25": {
                          "Number": 25,
                          "Score": 0.3333333333333333
                        },
                        "28": {
```

```
                        "Number": 28,
                        "Score": 0.5
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Kulczynski": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
.Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.8333333333333333
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(
System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Multiply(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(
System.Double,System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Divide(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Divide(
System.Double,System.Double)",
                      "Lines": {
```

```
                        "25": {
                          "Number": 25,
                          "Score": 0.4166666666666663
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.6666666666666666
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Rogers-Tanimoto": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
.Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.2
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(
System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Multiply(System.Double,
System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(
System.Double,System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    },
```

```
                    "System.Double Calculators.Calculator::Divide(System.Double,
    System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Divide(
    System.Double,System.Double)",
                      "Lines": {
                        "25": {
                          "Number": 25,
                          "Score": 0.2
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.09090909090909091
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Seban Normalization": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
    Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
    Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
    .Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 1.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,
    System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(
    System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Multiply(System.Double,
    System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(
    System.Double,System.Double)",
                      "Lines": {
                        "20": {
```

```
                        "Number": 20,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Double Calculators.Calculator::Divide(System.Double,
    System.Double)": {
                    "Signature": "System.Double Calculators.Calculator::Divide(
    System.Double,System.Double)",
                    "Lines": {
                      "25": {
                        "Number": 25,
                        "Score": 0.0
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.0
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Tarantula & Sampaio": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
    Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
    Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
    .Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 1.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,
    System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(
    System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
```

```
                    "System.Double Calculators.Calculator::Multiply(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(System.Double,System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Divide(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Divide(System.Double,System.Double)",
                      "Lines": {
                        "25": {
                          "Number": 25,
                          "Score": 0.6666666666666666
                        },
                        "28": {
                          "Number": 28,
                          "Score": 1.0
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Sokal Sneath & Sampaio": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System.Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.4
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Subtract(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(System.Double,System.Double)",
                      "Lines": {
                        "15": {
```

```
                        "Number": 15,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Double Calculators.Calculator::Multiply(System.Double,
    System.Double)": {
                    "Signature": "System.Double Calculators.Calculator::Multiply(
    System.Double,System.Double)",
                    "Lines": {
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Double Calculators.Calculator::Divide(System.Double,
    System.Double)": {
                    "Signature": "System.Double Calculators.Calculator::Divide(
    System.Double,System.Double)",
                    "Lines": {
                      "25": {
                        "Number": 25,
                        "Score": 0.36363636363636365
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.25
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Roger-Tanimoto & Sampaio": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\
    Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.
    Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System
    .Double,System.Double)",
                      "Lines": {
                        "10": {
                          "Number": 10,
                          "Score": 0.2857142857142857
                        }
                      }
                    },
```

```
                    "System.Double Calculators.Calculator::Subtract(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Subtract(System.Double,System.Double)",
                      "Lines": {
                        "15": {
                          "Number": 15,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Multiply(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Multiply(System.Double,System.Double)",
                      "Lines": {
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Double Calculators.Calculator::Divide(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Divide(System.Double,System.Double)",
                      "Lines": {
                        "25": {
                          "Number": 25,
                          "Score": 0.26666666666666666
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.15384615384615385
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Simple Matching & Sampaio": {
      "Assemblies": {
        "Calculators.dll": {
          "Name": "Calculators.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\Calculators\\Calculator.cs",
              "Classes": {
                "Calculators.Calculator": {
                  "Name": "Calculators.Calculator",
                  "Methods": {
                    "System.Double Calculators.Calculator::Add(System.Double,System.Double)": {
                      "Signature": "System.Double Calculators.Calculator::Add(System.Double,System.Double)",
                      "Lines": {
                        "10": {
```

```
                        "Number": 10,
                        "Score": 0.3333333333333333
                      }
                    }
                  },
                  "System.Double Calculators.Calculator::Subtract(System.Double,
    System.Double)": {
                    "Signature": "System.Double Calculators.Calculator::Subtract(
    System.Double,System.Double)",
                    "Lines": {
                      "15": {
                        "Number": 15,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Double Calculators.Calculator::Multiply(System.Double,
    System.Double)": {
                    "Signature": "System.Double Calculators.Calculator::Multiply(
    System.Double,System.Double)",
                    "Lines": {
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Double Calculators.Calculator::Divide(System.Double,
    System.Double)": {
                    "Signature": "System.Double Calculators.Calculator::Divide(
    System.Double,System.Double)",
                    "Lines": {
                      "25": {
                        "Number": 25,
                        "Score": 0.2857142857142857
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.1666666666666666
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```


Listing F.4: Defects Sample Project - Calculators - Suspiciousness Results from Fault Detector .NET CLI


```
{
  "Techniques": {
    "Tarantula": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
```

```
          "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs",
            "Classes": {
              "FizzBuzzProject.FizzBuzz": {
                "Name": "FizzBuzzProject.FizzBuzz",
                "Methods": {
                  "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)": {
                    "Signature": "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)",
                    "Lines": {
                      "13": {
                        "Number": 13,
                        "Score": 0.5
                      },
                      "15": {
                        "Number": 15,
                        "Score": 0.6000000000000001
                      },
                      "18": {
                        "Number": 18,
                        "Score": 0.4285714285714286
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      },
                      "23": {
                        "Number": 23,
                        "Score": 0.6000000000000001
                      },
                      "25": {
                        "Number": 25,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Ochiai": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
                  "Name": "FizzBuzzProject.FizzBuzz",
                  "Methods": {
                    "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)": {
                      "Signature": "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)",
                      "Lines": {
```

```
                        "13": {
                          "Number": 13,
                          "Score": 0.42640143271122083
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.35355339059327373
                        },
                        "18": {
                          "Number": 18,
                          "Score": 0.2672612419124244
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "23": {
                          "Number": 23,
                          "Score": 0.35355339059327373
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.7071067811865475
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Jaccard": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\
FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
                  "Name": "FizzBuzzProject.FizzBuzz",
                  "Methods": {
                    "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int3
2)": {
                      "Signature": "System.String FizzBuzzProject.FizzBuzz::
GetFizzBuzz(System.Int32)",
                      "Lines": {
                        "13": {
                          "Number": 13,
                          "Score": 0.18181818181818182
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.2
                        },
                        "18": {
                          "Number": 18,
                          "Score": 0.125
                        },
```

```
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "23": {
                          "Number": 23,
                          "Score": 0.2
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.5
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "D2 (DStar)": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\
    FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
                  "Name": "FizzBuzzProject.FizzBuzz",
                  "Methods": {
                    "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int3
    2)": {
                      "Signature": "System.String FizzBuzzProject.FizzBuzz::
    GetFizzBuzz(System.Int32)",
                      "Lines": {
                        "13": {
                          "Number": 13,
                          "Score": 0.4444444444444444
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.25
                        },
                        "18": {
                          "Number": 18,
                          "Score": 0.14285714285714285
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "23": {
                          "Number": 23,
                          "Score": 0.25
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.0
                        },
```

```
                        "29": {
                          "Number": 29,
                          "Score": 1.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      },
      "Kulczynski": {
        "Assemblies": {
          "FizzBuzzProject.dll": {
            "Name": "FizzBuzzProject.dll",
            "Files": {
              "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
                "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\
FizzBuzz.cs",
                "Classes": {
                  "FizzBuzzProject.FizzBuzz": {
                    "Name": "FizzBuzzProject.FizzBuzz",
                    "Methods": {
                      "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int3
2)": {
                        "Signature": "System.String FizzBuzzProject.FizzBuzz::
GetFizzBuzz(System.Int32)",
                        "Lines": {
                          "13": {
                            "Number": 13,
                            "Score": 0.5909090909090909
                          },
                          "15": {
                            "Number": 15,
                            "Score": 0.375
                          },
                          "18": {
                            "Number": 18,
                            "Score": 0.3214285714285714
                          },
                          "20": {
                            "Number": 20,
                            "Score": 0.0
                          },
                          "23": {
                            "Number": 23,
                            "Score": 0.375
                          },
                          "25": {
                            "Number": 25,
                            "Score": 0.0
                          },
                          "29": {
                            "Number": 29,
                            "Score": 0.75
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
```

```
      }
    },
    "Rogers-Tanimoto": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
                  "Name": "FizzBuzzProject.FizzBuzz",
                  "Methods": {
                    "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)": {
                      "Signature": "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)",
                      "Lines": {
                        "13": {
                          "Number": 13,
                          "Score": 1.0
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.2222222222222222
                        },
                        "18": {
                          "Number": 18,
                          "Score": 0.4666666666666667
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "23": {
                          "Number": 23,
                          "Score": 0.2222222222222222
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.047619047619047616
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Seban Normalization": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
```

```
                "Name": "FizzBuzzProject.FizzBuzz",
                "Methods": {
                  "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)": {
                    "Signature": "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)",
                    "Lines": {
                      "13": {
                        "Number": 13,
                        "Score": 1.0
                      },
                      "15": {
                        "Number": 15,
                        "Score": 0.0
                      },
                      "18": {
                        "Number": 18,
                        "Score": 0.0
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      },
                      "23": {
                        "Number": 23,
                        "Score": 0.0
                      },
                      "25": {
                        "Number": 25,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  },
  "Tarantula & Sampaio": {
    "Assemblies": {
      "FizzBuzzProject.dll": {
        "Name": "FizzBuzzProject.dll",
        "Files": {
          "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs",
            "Classes": {
              "FizzBuzzProject.FizzBuzz": {
                "Name": "FizzBuzzProject.FizzBuzz",
                "Methods": {
                  "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)": {
                    "Signature": "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int32)",
                    "Lines": {
                      "13": {
                        "Number": 13,
                        "Score": 0.5499999999999999
                      },
                      "15": {
```

```
                        "Number": 15,
                        "Score": 0.6666666666666666
                      },
                      "18": {
                        "Number": 18,
                        "Score": 0.4666666666666667
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      },
                      "23": {
                        "Number": 23,
                        "Score": 0.6666666666666666
                      },
                      "25": {
                        "Number": 25,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 1.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Sokal Sneath & Sampaio": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\
FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
                  "Name": "FizzBuzzProject.FizzBuzz",
                  "Methods": {
                    "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int3
2)": {
                      "Signature": "System.String FizzBuzzProject.FizzBuzz::
GetFizzBuzz(System.Int32)",
                      "Lines": {
                        "13": {
                          "Number": 13,
                          "Score": 0.5238095238095238
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.36363636363636365
                        },
                        "18": {
                          "Number": 18,
                          "Score": 0.45161290322580644
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "23": {
```

```
                        "Number": 23,
                        "Score": 0.36363636363636365
                      },
                      "25": {
                        "Number": 25,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.15384615384615385
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Roger-Tanimoto & Sampaio": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\
    FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
                  "Name": "FizzBuzzProject.FizzBuzz",
                  "Methods": {
                    "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int3
    2)": {
                      "Signature": "System.String FizzBuzzProject.FizzBuzz::
    GetFizzBuzz(System.Int32)",
                      "Lines": {
                        "13": {
                          "Number": 13,
                          "Score": 0.5238095238095238
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.27586206896551724
                        },
                        "18": {
                          "Number": 18,
                          "Score": 0.4
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "23": {
                          "Number": 23,
                          "Score": 0.27586206896551724
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.08695652173913043
                        }
                      }
```

```
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Simple Matching & Sampaio": {
      "Assemblies": {
        "FizzBuzzProject.dll": {
          "Name": "FizzBuzzProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\FizzBuzz.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\FizzBuzzProject\\
    FizzBuzz.cs",
              "Classes": {
                "FizzBuzzProject.FizzBuzz": {
                  "Name": "FizzBuzzProject.FizzBuzz",
                  "Methods": {
                    "System.String FizzBuzzProject.FizzBuzz::GetFizzBuzz(System.Int3
    2)": {
                      "Signature": "System.String FizzBuzzProject.FizzBuzz::
    GetFizzBuzz(System.Int32)",
                      "Lines": {
                        "13": {
                          "Number": 13,
                          "Score": 0.55
                        },
                        "15": {
                          "Number": 15,
                          "Score": 0.2857142857142857
                        },
                        "18": {
                          "Number": 18,
                          "Score": 0.4117647058823529
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        },
                        "23": {
                          "Number": 23,
                          "Score": 0.2857142857142857
                        },
                        "25": {
                          "Number": 25,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.09090909090909091
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Listing F.5: Defects Sample Project - FizzBuzzProject - Suspiciousness Results from Fault Detector .NET CLI


```
{
  "Techniques": {
    "Tarantula": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs" : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 0.5
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.6000000000000001
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Ochiai": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs" : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
```

```
                "Methods": {
                  "System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)": {
                    "Signature": "System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)",
                    "Lines": {
                      "9": {
                        "Number": 9,
                        "Score": 0.5773502691896257
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.0
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.6546536707079772
                      },
                      "17": {
                        "Number": 17,
                        "Score": 1.0
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Jaccard": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 0.3333333333333333
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.42857142857142855
                        },
                        "17": {
```

```
                        "Number": 17,
                        "Score": 1.0
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "D2 (DStar)": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 1.5
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 2.25
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.0
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Kulczynski": {
      "Assemblies": {
```

```
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 0.6666666666666666
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.7142857142857143
                        },
                        "17": {
                          "Number": 17,
                          "Score": 1.0
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      },
      "Rogers-Tanimoto": {
        "Assemblies": {
          "MaxFloatProject.dll": {
            "Name": "MaxFloatProject.dll",
            "Files": {
              "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 1.0
```

```
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.6363636363636364
                        },
                        "17": {
                          "Number": 17,
                          "Score": 0.2
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Seban Normalization": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 0.0
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.0
                        },
                        "17": {
                          "Number": 17,
                          "Score": 1.0
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    }
```

```
                }
              }
            }
          }
        }
      }
    },
    "Tarantula & Sampaio": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 0.6000000000000001
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.7241379310344828
                        },
                        "17": {
                          "Number": 17,
                          "Score": 1.0
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Sokal Sneath & Sampaio": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
```

```
                "Name": "MaxFloatProject.MaxFloatProject",
                "Methods": {
                  "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                    "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                    "Lines": {
                      "9": {
                        "Number": 9,
                        "Score": 0.5454545454545454
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.0
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.5185185185185185
                      },
                      "17": {
                        "Number": 17,
                        "Score": 0.4
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Roger-Tanimoto & Sampaio": {
      "Assemblies": {
        "MaxFloatProject.dll": {
          "Name": "MaxFloatProject.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
              "Classes": {
                "MaxFloatProject.MaxFloatProject": {
                  "Name": "MaxFloatProject.MaxFloatProject",
                  "Methods": {
                    "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                      "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                      "Lines": {
                        "9": {
                          "Number": 9,
                          "Score": 0.5454545454545454
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.0
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.4827586206896552
                        },
```

```
                        "17": {
                          "Number": 17,
                          "Score": 0.2857142857142857
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.0
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      },
      "Simple Matching & Sampaio": {
        "Assemblies": {
          "MaxFloatProject.dll": {
            "Name": "MaxFloatProject.dll",
            "Files": {
              "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\MaxFloatProject.cs"
    : {
                "SourcePath": "C:\\Project\\DefectsSampleProject\\MaxFloatProject\\
    MaxFloatProject.cs",
                "Classes": {
                  "MaxFloatProject.MaxFloatProject": {
                    "Name": "MaxFloatProject.MaxFloatProject",
                    "Methods": {
                      "System.Single MaxFloatProject.MaxFloatProject::Max(System.
    Single,System.Single)": {
                        "Signature": "System.Single MaxFloatProject.MaxFloatProject::
    Max(System.Single,System.Single)",
                        "Lines": {
                          "9": {
                            "Number": 9,
                            "Score": 0.6
                          },
                          "11": {
                            "Number": 11,
                            "Score": 0.0
                          },
                          "14": {
                            "Number": 14,
                            "Score": 0.5384615384615384
                          },
                          "17": {
                            "Number": 17,
                            "Score": 0.3333333333333333
                          },
                          "20": {
                            "Number": 20,
                            "Score": 0.0
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
```

```
}
```

Listing F.6: Defects Sample Project - MaxFloatProject - Suspiciousness Results from Fault Detector .NET CLI


```
{
  "Techniques": {
    "Tarantula": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs",
              "Classes": {
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                      "Signature": "System.Boolean TaskManagers.Task::get_IsCompleted()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Category()": {
                      "Signature": "System.String TaskManagers.Task::get_Category()",
                      "Lines": {}
                    },
                    "System.Int32 TaskManagers.Task::get_Priority()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                      "Lines": {}
                    },
                    "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.0
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.0
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.0
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.0
                        }
```

```
                    }
                  },
                  "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
,System.String,System.String,System.Boolean,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                    "Lines": {
                      "5": {
                        "Number": 5,
                        "Score": 0.6666666666666666
                      },
                      "7": {
                        "Number": 7,
                        "Score": 0.6666666666666666
                      },
                      "9": {
                        "Number": 9,
                        "Score": 0.6666666666666666
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.6666666666666666
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.6666666666666666
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.6666666666666666
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.6666666666666666
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
TaskManager.cs",
            "Classes": {
              "TaskManagers.TaskManager": {
                "Name": "TaskManagers.TaskManager",
                "Methods": {
                  "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
String,System.String,System.String,System.Int32)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 0.6666666666666666
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.6666666666666666
                      },
                      "21": {
                        "Number": 21,
                        "Score": 0.6666666666666666
                      }
                    }
                  },
```

```
                    "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.0
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.0
                        },
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)",
                      "Lines": {
                        "38": {
                          "Number": 38,
                          "Score": 0.27272727272727276
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()",
                      "Lines": {
                        "43": {
                          "Number": 43,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
                          "Number": 54,
```

```
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "59": {
                        "Number": 59,
                        "Score": 0.6000000000000001
                      },
                      "60": {
                        "Number": 60,
                        "Score": 0.6000000000000001
                      },
                      "62": {
                        "Number": 62,
                        "Score": 0.0
                      },
                      "65": {
                        "Number": 65,
                        "Score": 0.0
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void TaskManagers.TaskManager::.ctor()": {
                    "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                    "Lines": {
                      "11": {
                        "Number": 11,
                        "Score": 0.5
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.5
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.5
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Ochiai": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs",
              "Classes": {
                "TaskManagers.Task": {
```

```
                "Name": "TaskManagers.Task",
                "Methods": {
                  "System.Int32 TaskManagers.Task::get_Id()": {
                    "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Title()": {
                    "Signature": "System.String TaskManagers.Task::get_Title()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Description()": {
                    "Signature": "System.String TaskManagers.Task::get_Description
    ()",
                    "Lines": {}
                  },
                  "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                    "Signature": "System.Boolean TaskManagers.Task::
    get_IsCompleted()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Category()": {
                    "Signature": "System.String TaskManagers.Task::get_Category()"
    ,
                    "Lines": {}
                  },
                  "System.Int32 TaskManagers.Task::get_Priority()": {
                    "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                    "Lines": {}
                  },
                  "System.Void TaskManagers.Task::Update(System.String,System.
    String,System.String,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::Update(System.
    String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.5
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.5
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.5
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.5
                      }
                    }
                  },
                  "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
    ,System.String,System.String,System.Boolean,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
    2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                    "Lines": {
                      "5": {
                        "Number": 5,
                        "Score": 0.7559289460184544
                      },
                      "7": {
                        "Number": 7,
                        "Score": 0.7559289460184544
                      },
                      "9": {
                        "Number": 9,
```

```
                        "Score": 0.7559289460184544
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.7559289460184544
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.7559289460184544
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.7559289460184544
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.7559289460184544
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
    TaskManager.cs",
            "Classes": {
              "TaskManagers.TaskManager": {
                "Name": "TaskManagers.TaskManager",
                "Methods": {
                  "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
    String,System.String,System.String,System.Int32)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 0.7559289460184544
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.7559289460184544
                      },
                      "21": {
                        "Number": 21,
                        "Score": 0.7559289460184544
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
    2)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::
    RemoveTask(System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.0
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
                      },
```

```
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)",
                      "Lines": {
                        "38": {
                          "Number": 38,
                          "Score": 0.22360679774997896
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()",
                      "Lines": {
                        "43": {
                          "Number": 43,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 0.5
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
                          "Number": 54,
                          "Score": 0.5
                        }
                      }
                    },
                    "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "59": {
                          "Number": 59,
                          "Score": 0.35355339059327373
                        },
                        "60": {
                          "Number": 60,
                          "Score": 0.35355339059327373
```

```
                        },
                        "62": {
                          "Number": 62,
                          "Score": 0.0
                        },
                        "65": {
                          "Number": 65,
                          "Score": 0.5
                        },
                        "66": {
                          "Number": 66,
                          "Score": 0.5
                        }
                      }
                    },
                    "System.Void TaskManagers.TaskManager::.ctor()": {
                      "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                      "Lines": {
                        "11": {
                          "Number": 11,
                          "Score": 0.6324555320336759
                        },
                        "13": {
                          "Number": 13,
                          "Score": 0.6324555320336759
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.6324555320336759
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Jaccard": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.
cs",
              "Classes": {
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description
()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
```

```
                    "Signature": "System.Boolean TaskManagers.Task::
    get_IsCompleted()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Category()": {
                    "Signature": "System.String TaskManagers.Task::get_Category()"
    ,
                    "Lines": {}
                  },
                  "System.Int32 TaskManagers.Task::get_Priority()": {
                    "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                    "Lines": {}
                  },
                  "System.Void TaskManagers.Task::Update(System.String,System.
    String,System.String,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::Update(System.
    String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.25
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.25
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.25
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.25
                      }
                    }
                  },
                  "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
    ,System.String,System.String,System.Boolean,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
    2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                    "Lines": {
                      "5": {
                        "Number": 5,
                        "Score": 0.5714285714285714
                      },
                      "7": {
                        "Number": 7,
                        "Score": 0.5714285714285714
                      },
                      "9": {
                        "Number": 9,
                        "Score": 0.5714285714285714
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.5714285714285714
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.5714285714285714
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.5714285714285714
                      },
                      "14": {
                        "Number": 14,
```

```
                        "Score": 0.5714285714285714
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
    TaskManager.cs",
              "Classes": {
                "TaskManagers.TaskManager": {
                  "Name": "TaskManagers.TaskManager",
                  "Methods": {
                    "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
    String,System.String,System.String,System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    AddTask(System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "19": {
                          "Number": 19,
                          "Score": 0.5714285714285714
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.5714285714285714
                        },
                        "21": {
                          "Number": 21,
                          "Score": 0.5714285714285714
                        }
                      }
                    },
                    "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
    2)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::
    RemoveTask(System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.0
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.0
                        },
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int3
    2)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    GetTask(System.Int32)",
                      "Lines": {
                        "38": {
```

```
                        "Number": 38,
                        "Score": 0.125
                      }
                    }
                  },
                  "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetAllTasks()": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetAllTasks()",
                  "Lines": {
                    "43": {
                      "Number": 43,
                      "Score": 0.0
                    }
                  }
                },
                "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                  "Lines": {
                    "48": {
                      "Number": 48,
                      "Score": 0.25
                    }
                  }
                },
                "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                  "Lines": {
                    "54": {
                      "Number": 54,
                      "Score": 0.25
                    }
                  }
                },
                "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int3
    2,System.String,System.String,System.String,System.Int32)": {
                  "Signature": "System.Boolean TaskManagers.TaskManager::
    UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)"
    ,
                  "Lines": {
                    "59": {
                      "Number": 59,
                      "Score": 0.2
                    },
                    "60": {
                      "Number": 60,
                      "Score": 0.2
                    },
                    "62": {
                      "Number": 62,
                      "Score": 0.0
                    },
                    "65": {
                      "Number": 65,
                      "Score": 0.25
                    },
                    "66": {
                      "Number": 66,
                      "Score": 0.25
                    }
                  }
                },
                "System.Void TaskManagers.TaskManager::.ctor()": {
```

```
                    "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                    "Lines": {
                      "11": {
                        "Number": 11,
                        "Score": 0.4
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.4
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.4
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "D2 (DStar)": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs",
              "Classes": {
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                      "Signature": "System.Boolean TaskManagers.Task::get_IsCompleted()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Category()": {
                      "Signature": "System.String TaskManagers.Task::get_Category()",
                      "Lines": {}
                    },
                    "System.Int32 TaskManagers.Task::get_Priority()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                      "Lines": {}
                    },
                    "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)",
```

```
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.3333333333333333
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.3333333333333333
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.3333333333333333
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.3333333333333333
                      }
                    }
                  },
                  "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
,System.String,System.String,System.Boolean,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                    "Lines": {
                      "5": {
                        "Number": 5,
                        "Score": 5.333333333333333
                      },
                      "7": {
                        "Number": 7,
                        "Score": 5.333333333333333
                      },
                      "9": {
                        "Number": 9,
                        "Score": 5.333333333333333
                      },
                      "10": {
                        "Number": 10,
                        "Score": 5.333333333333333
                      },
                      "11": {
                        "Number": 11,
                        "Score": 5.333333333333333
                      },
                      "13": {
                        "Number": 13,
                        "Score": 5.333333333333333
                      },
                      "14": {
                        "Number": 14,
                        "Score": 5.333333333333333
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
TaskManager.cs",
            "Classes": {
              "TaskManagers.TaskManager": {
                "Name": "TaskManagers.TaskManager",
                "Methods": {
                  "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
String,System.String,System.String,System.Int32)": {
```

```
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 5.333333333333333
                      },
                      "20": {
                        "Number": 20,
                        "Score": 5.333333333333333
                      },
                      "21": {
                        "Number": 21,
                        "Score": 5.333333333333333
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
2)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::
RemoveTask(System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.0
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
                      },
                      "32": {
                        "Number": 32,
                        "Score": 0.0
                      },
                      "33": {
                        "Number": 33,
                        "Score": 0.0
                      }
                    }
                  },
                  "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int3
2)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
GetTask(System.Int32)",
                    "Lines": {
                      "38": {
                        "Number": 38,
                        "Score": 0.14285714285714285
                      }
                    }
                  },
                  "System.Collections.Generic.List`1<TaskManagers.Task>
TaskManagers.TaskManager::GetAllTasks()": {
                    "Signature": "System.Collections.Generic.List`1<TaskManagers.
Task> TaskManagers.TaskManager::GetAllTasks()",
                    "Lines": {
                      "43": {
                        "Number": 43,
                        "Score": 0.0
                      }
                    }
                  },
```

```
                    "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 0.3333333333333333
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
                          "Number": 54,
                          "Score": 0.3333333333333333
                        }
                      }
                    },
                    "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int3
    2,System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::
    UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)"
    ,
                      "Lines": {
                        "59": {
                          "Number": 59,
                          "Score": 0.25
                        },
                        "60": {
                          "Number": 60,
                          "Score": 0.25
                        },
                        "62": {
                          "Number": 62,
                          "Score": 0.0
                        },
                        "65": {
                          "Number": 65,
                          "Score": 0.3333333333333333
                        },
                        "66": {
                          "Number": 66,
                          "Score": 0.3333333333333333
                        }
                      }
                    },
                    "System.Void TaskManagers.TaskManager::.ctor()": {
                      "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                      "Lines": {
                        "11": {
                          "Number": 11,
                          "Score": 2.6666666666666665
                        },
                        "13": {
                          "Number": 13,
                          "Score": 2.6666666666666665
                        },
                        "14": {
                          "Number": 14,
                          "Score": 2.6666666666666665
                        }
                      }
                    }
```

```
                }
              }
            }
          }
        }
      }
    },
    "Kulczynski": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs",
              "Classes": {
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                      "Signature": "System.Boolean TaskManagers.Task::get_IsCompleted()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Category()": {
                      "Signature": "System.String TaskManagers.Task::get_Category()",
                      "Lines": {}
                    },
                    "System.Int32 TaskManagers.Task::get_Priority()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                      "Lines": {}
                    },
                    "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.625
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.625
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.625
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.625
```

```
                        }
                      }
                    },
                    "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
,System.String,System.String,System.Boolean,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                      "Lines": {
                        "5": {
                          "Number": 5,
                          "Score": 0.7857142857142857
                        },
                        "7": {
                          "Number": 7,
                          "Score": 0.7857142857142857
                        },
                        "9": {
                          "Number": 9,
                          "Score": 0.7857142857142857
                        },
                        "10": {
                          "Number": 10,
                          "Score": 0.7857142857142857
                        },
                        "11": {
                          "Number": 11,
                          "Score": 0.7857142857142857
                        },
                        "13": {
                          "Number": 13,
                          "Score": 0.7857142857142857
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.7857142857142857
                        }
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
TaskManager.cs",
              "Classes": {
                "TaskManagers.TaskManager": {
                  "Name": "TaskManagers.TaskManager",
                  "Methods": {
                    "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
String,System.String,System.String,System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::
AddTask(System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "19": {
                          "Number": 19,
                          "Score": 0.7857142857142857
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.7857142857142857
                        },
                        "21": {
                          "Number": 21,
                          "Score": 0.7857142857142857
                        }
                      }
```

```
                    },
                    "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.0
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.0
                        },
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)",
                      "Lines": {
                        "38": {
                          "Number": 38,
                          "Score": 0.225
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()",
                      "Lines": {
                        "43": {
                          "Number": 43,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 0.625
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
```

```
                        "Number": 54,
                        "Score": 0.625
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "59": {
                        "Number": 59,
                        "Score": 0.375
                      },
                      "60": {
                        "Number": 60,
                        "Score": 0.375
                      },
                      "62": {
                        "Number": 62,
                        "Score": 0.0
                      },
                      "65": {
                        "Number": 65,
                        "Score": 0.625
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.625
                      }
                    }
                  },
                  "System.Void TaskManagers.TaskManager::.ctor()": {
                    "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                    "Lines": {
                      "11": {
                        "Number": 11,
                        "Score": 0.7
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.7
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.7
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Rogers-Tanimoto": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs",
              "Classes": {
```

```
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description
    ()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                      "Signature": "System.Boolean TaskManagers.Task::
    get_IsCompleted()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Category()": {
                      "Signature": "System.String TaskManagers.Task::get_Category()"
    ,
                      "Lines": {}
                    },
                    "System.Int32 TaskManagers.Task::get_Priority()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                      "Lines": {}
                    },
                    "System.Void TaskManagers.Task::Update(System.String,System.
    String,System.String,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::Update(System.
    String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.05263157894736842
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.05263157894736842
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.05263157894736842
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.05263157894736842
                        }
                      }
                    },
                    "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
    ,System.String,System.String,System.Boolean,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
    2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                      "Lines": {
                        "5": {
                          "Number": 5,
                          "Score": 0.5384615384615384
                        },
                        "7": {
                          "Number": 7,
                          "Score": 0.5384615384615384
                        },
                        "9": {
```

```
                        "Number": 9,
                        "Score": 0.5384615384615384
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.5384615384615384
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.5384615384615384
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.5384615384615384
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.5384615384615384
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
    TaskManager.cs",
            "Classes": {
              "TaskManagers.TaskManager": {
                "Name": "TaskManagers.TaskManager",
                "Methods": {
                  "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
    String,System.String,System.String,System.Int32)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 0.5384615384615384
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.5384615384615384
                      },
                      "21": {
                        "Number": 21,
                        "Score": 0.5384615384615384
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
    2)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::
    RemoveTask(System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.0
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
```

```
                        },
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)",
                      "Lines": {
                        "38": {
                          "Number": 38,
                          "Score": 0.3333333333333333
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()",
                      "Lines": {
                        "43": {
                          "Number": 43,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 0.05263157894736842
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
                          "Number": 54,
                          "Score": 0.05263157894736842
                        }
                      }
                    },
                    "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "59": {
                          "Number": 59,
                          "Score": 0.1111111111111111
                        },
                        "60": {
                          "Number": 60,
```

```
                        "Score": 0.1111111111111111
                      },
                      "62": {
                        "Number": 62,
                        "Score": 0.0
                      },
                      "65": {
                        "Number": 65,
                        "Score": 0.05263157894736842
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.05263157894736842
                      }
                    }
                  },
                  "System.Void TaskManagers.TaskManager::.ctor()": {
                    "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                    "Lines": {
                      "11": {
                        "Number": 11,
                        "Score": 1.0
                      },
                      "13": {
                        "Number": 13,
                        "Score": 1.0
                      },
                      "14": {
                        "Number": 14,
                        "Score": 1.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Seban Normalization": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.
    cs",
              "Classes": {
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description
    ()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
```

```
                    "Signature": "System.Boolean TaskManagers.Task::
    get_IsCompleted()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Category()": {
                    "Signature": "System.String TaskManagers.Task::get_Category()"
    ,
                    "Lines": {}
                  },
                  "System.Int32 TaskManagers.Task::get_Priority()": {
                    "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                    "Lines": {}
                  },
                  "System.Void TaskManagers.Task::Update(System.String,System.
    String,System.String,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::Update(System.
    String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.0
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.0
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.0
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
    ,System.String,System.String,System.Boolean,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
    2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                    "Lines": {
                      "5": {
                        "Number": 5,
                        "Score": 1.0
                      },
                      "7": {
                        "Number": 7,
                        "Score": 1.0
                      },
                      "9": {
                        "Number": 9,
                        "Score": 1.0
                      },
                      "10": {
                        "Number": 10,
                        "Score": 1.0
                      },
                      "11": {
                        "Number": 11,
                        "Score": 1.0
                      },
                      "13": {
                        "Number": 13,
                        "Score": 1.0
                      },
                      "14": {
                        "Number": 14,
```

```
                        "Score": 1.0
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
    TaskManager.cs",
            "Classes": {
              "TaskManagers.TaskManager": {
                "Name": "TaskManagers.TaskManager",
                "Methods": {
                  "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
    String,System.String,System.String,System.Int32)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 1.0
                      },
                      "20": {
                        "Number": 20,
                        "Score": 1.0
                      },
                      "21": {
                        "Number": 21,
                        "Score": 1.0
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
    2)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::
    RemoveTask(System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.0
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
                      },
                      "32": {
                        "Number": 32,
                        "Score": 0.0
                      },
                      "33": {
                        "Number": 33,
                        "Score": 0.0
                      }
                    }
                  },
                  "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int3
    2)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    GetTask(System.Int32)",
                    "Lines": {
                      "38": {
```

```
                        "Number": 38,
                        "Score": 0.0
                      }
                    }
                  },
                  "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetAllTasks()": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetAllTasks()",
                  "Lines": {
                    "43": {
                      "Number": 43,
                      "Score": 0.0
                    }
                  }
                },
                "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                  "Lines": {
                    "48": {
                      "Number": 48,
                      "Score": 0.0
                    }
                  }
                },
                "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                  "Lines": {
                    "54": {
                      "Number": 54,
                      "Score": 0.0
                    }
                  }
                },
                "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int3
    2,System.String,System.String,System.String,System.Int32)": {
                  "Signature": "System.Boolean TaskManagers.TaskManager::
    UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)"
    ,
                  "Lines": {
                    "59": {
                      "Number": 59,
                      "Score": 0.0
                    },
                    "60": {
                      "Number": 60,
                      "Score": 0.0
                    },
                    "62": {
                      "Number": 62,
                      "Score": 0.0
                    },
                    "65": {
                      "Number": 65,
                      "Score": 0.0
                    },
                    "66": {
                      "Number": 66,
                      "Score": 0.0
                    }
                  }
                },
                "System.Void TaskManagers.TaskManager::.ctor()": {
```

```
                    "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                    "Lines": {
                      "11": {
                        "Number": 11,
                        "Score": 0.0
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.0
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.0
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  },
  "Tarantula & Sampaio": {
    "Assemblies": {
      "TaskManagers.dll": {
        "Name": "TaskManagers.dll",
        "Files": {
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs",
            "Classes": {
              "TaskManagers.Task": {
                "Name": "TaskManagers.Task",
                "Methods": {
                  "System.Int32 TaskManagers.Task::get_Id()": {
                    "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Title()": {
                    "Signature": "System.String TaskManagers.Task::get_Title()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Description()": {
                    "Signature": "System.String TaskManagers.Task::get_Description()",
                    "Lines": {}
                  },
                  "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                    "Signature": "System.Boolean TaskManagers.Task::get_IsCompleted()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Category()": {
                    "Signature": "System.String TaskManagers.Task::get_Category()",
                    "Lines": {}
                  },
                  "System.Int32 TaskManagers.Task::get_Priority()": {
                    "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                    "Lines": {}
                  },
                  "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::Update(System.String,System.String,System.String,System.Int32)",
```

```
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 1.0
                      },
                      "28": {
                        "Number": 28,
                        "Score": 1.0
                      },
                      "30": {
                        "Number": 30,
                        "Score": 1.0
                      },
                      "31": {
                        "Number": 31,
                        "Score": 1.0
                      }
                    }
                  },
                  "System.Void TaskManagers.Task::.ctor(System.Int32,System.String,System.String,System.String,System.Boolean,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::.ctor(System.Int32,System.String,System.String,System.String,System.Boolean,System.Int32)",
                    "Lines": {
                      "5": {
                        "Number": 5,
                        "Score": 0.823529411764706
                      },
                      "7": {
                        "Number": 7,
                        "Score": 0.823529411764706
                      },
                      "9": {
                        "Number": 9,
                        "Score": 0.823529411764706
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.823529411764706
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.823529411764706
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.823529411764706
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.823529411764706
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs",
              "Classes": {
                "TaskManagers.TaskManager": {
                  "Name": "TaskManagers.TaskManager",
                  "Methods": {
                    "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.String,System.String,System.String,System.Int32)": {
```

```
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 0.823529411764706
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.823529411764706
                      },
                      "21": {
                        "Number": 21,
                        "Score": 0.823529411764706
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
    2)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::
    RemoveTask(System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.0
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
                      },
                      "32": {
                        "Number": 32,
                        "Score": 0.0
                      },
                      "33": {
                        "Number": 33,
                        "Score": 0.0
                      }
                    }
                  },
                  "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int3
    2)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    GetTask(System.Int32)",
                    "Lines": {
                      "38": {
                        "Number": 38,
                        "Score": 0.3191489361702128
                      }
                    }
                  },
                  "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetAllTasks()": {
                    "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetAllTasks()",
                    "Lines": {
                      "43": {
                        "Number": 43,
                        "Score": 0.0
                      }
                    }
                  },
```

```
                    "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 1.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
                          "Number": 54,
                          "Score": 1.0
                        }
                      }
                    },
                    "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int3
    2,System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::
    UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)"
    ,
                      "Lines": {
                        "59": {
                          "Number": 59,
                          "Score": 0.75
                        },
                        "60": {
                          "Number": 60,
                          "Score": 0.75
                        },
                        "62": {
                          "Number": 62,
                          "Score": 0.0
                        },
                        "65": {
                          "Number": 65,
                          "Score": 1.0
                        },
                        "66": {
                          "Number": 66,
                          "Score": 1.0
                        }
                      }
                    },
                    "System.Void TaskManagers.TaskManager::.ctor()": {
                      "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                      "Lines": {
                        "11": {
                          "Number": 11,
                          "Score": 0.625
                        },
                        "13": {
                          "Number": 13,
                          "Score": 0.625
                        },
                        "14": {
                          "Number": 14,
                          "Score": 0.625
                        }
                      }
                    }
```

```
                  }
                }
              }
            }
          }
        }
      }
    },
    "Sokal Sneath & Sampaio": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.
    cs",
              "Classes": {
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description
    ()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                      "Signature": "System.Boolean TaskManagers.Task::
    get_IsCompleted()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Category()": {
                      "Signature": "System.String TaskManagers.Task::get_Category()"
    ,
                      "Lines": {}
                    },
                    "System.Int32 TaskManagers.Task::get_Priority()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                      "Lines": {}
                    },
                    "System.Void TaskManagers.Task::Update(System.String,System.
    String,System.String,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::Update(System.
    String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.16666666666666666
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.16666666666666666
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.16666666666666666
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.16666666666666666
```

```
                    }
                  }
                },
                "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
,System.String,System.String,System.Boolean,System.Int32)": {
                  "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                  "Lines": {
                    "5": {
                      "Number": 5,
                      "Score": 0.5185185185185185
                    },
                    "7": {
                      "Number": 7,
                      "Score": 0.5185185185185185
                    },
                    "9": {
                      "Number": 9,
                      "Score": 0.5185185185185185
                    },
                    "10": {
                      "Number": 10,
                      "Score": 0.5185185185185185
                    },
                    "11": {
                      "Number": 11,
                      "Score": 0.5185185185185185
                    },
                    "13": {
                      "Number": 13,
                      "Score": 0.5185185185185185
                    },
                    "14": {
                      "Number": 14,
                      "Score": 0.5185185185185185
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
TaskManager.cs",
            "Classes": {
              "TaskManagers.TaskManager": {
                "Name": "TaskManagers.TaskManager",
                "Methods": {
                  "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
String,System.String,System.String,System.Int32)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 0.5185185185185185
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.5185185185185185
                      },
                      "21": {
                        "Number": 21,
                        "Score": 0.5185185185185185
                      }
                    }
```

```
                    },
                    "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.0
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.0
                        },
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)",
                      "Lines": {
                        "38": {
                          "Number": 38,
                          "Score": 0.4166666666666667
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()",
                      "Lines": {
                        "43": {
                          "Number": 43,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 0.1666666666666666
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
```

```
                        "Number": 54,
                        "Score": 0.16666666666666666
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "59": {
                        "Number": 59,
                        "Score": 0.26666666666666666
                      },
                      "60": {
                        "Number": 60,
                        "Score": 0.26666666666666666
                      },
                      "62": {
                        "Number": 62,
                        "Score": 0.0
                      },
                      "65": {
                        "Number": 65,
                        "Score": 0.16666666666666666
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.16666666666666666
                      }
                    }
                  },
                  "System.Void TaskManagers.TaskManager::.ctor()": {
                    "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                    "Lines": {
                      "11": {
                        "Number": 11,
                        "Score": 0.5555555555555556
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.5555555555555556
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.5555555555555556
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Roger-Tanimoto & Sampaio": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs",
              "Classes": {
```

```
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description
    ()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
                      "Signature": "System.Boolean TaskManagers.Task::
    get_IsCompleted()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Category()": {
                      "Signature": "System.String TaskManagers.Task::get_Category()"
    ,
                      "Lines": {}
                    },
                    "System.Int32 TaskManagers.Task::get_Priority()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                      "Lines": {}
                    },
                    "System.Void TaskManagers.Task::Update(System.String,System.
    String,System.String,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::Update(System.
    String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.09523809523809523
                        },
                        "28": {
                          "Number": 28,
                          "Score": 0.09523809523809523
                        },
                        "30": {
                          "Number": 30,
                          "Score": 0.09523809523809523
                        },
                        "31": {
                          "Number": 31,
                          "Score": 0.09523809523809523
                        }
                      }
                    },
                    "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
    ,System.String,System.String,System.Boolean,System.Int32)": {
                      "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
    2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                      "Lines": {
                        "5": {
                          "Number": 5,
                          "Score": 0.4666666666666667
                        },
                        "7": {
                          "Number": 7,
                          "Score": 0.4666666666666667
                        },
                        "9": {
```

```
                        "Number": 9,
                        "Score": 0.4666666666666667
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.4666666666666667
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.4666666666666667
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.4666666666666667
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.4666666666666667
                      }
                    }
                  }
                }
              }
            }
          },
          "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
            "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
    TaskManager.cs",
            "Classes": {
              "TaskManagers.TaskManager": {
                "Name": "TaskManagers.TaskManager",
                "Methods": {
                  "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
    String,System.String,System.String,System.Int32)": {
                    "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    AddTask(System.String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "19": {
                        "Number": 19,
                        "Score": 0.4666666666666667
                      },
                      "20": {
                        "Number": 20,
                        "Score": 0.4666666666666667
                      },
                      "21": {
                        "Number": 21,
                        "Score": 0.4666666666666667
                      }
                    }
                  },
                  "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
    2)": {
                    "Signature": "System.Boolean TaskManagers.TaskManager::
    RemoveTask(System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.0
                      },
                      "27": {
                        "Number": 27,
                        "Score": 0.0
                      },
                      "29": {
                        "Number": 29,
                        "Score": 0.0
```

```
                        },
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int32)",
                      "Lines": {
                        "38": {
                          "Number": 38,
                          "Score": 0.3448275862068966
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetAllTasks()",
                      "Lines": {
                        "43": {
                          "Number": 43,
                          "Score": 0.0
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                      "Lines": {
                        "48": {
                          "Number": 48,
                          "Score": 0.09523809523809523
                        }
                      }
                    },
                    "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                      "Signature": "System.Collections.Generic.List`1<TaskManagers.Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                      "Lines": {
                        "54": {
                          "Number": 54,
                          "Score": 0.09523809523809523
                        }
                      }
                    },
                    "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "59": {
                          "Number": 59,
                          "Score": 0.17391304347826086
                        },
                        "60": {
                          "Number": 60,
```

```
                        "Score": 0.17391304347826086
                      },
                      "62": {
                        "Number": 62,
                        "Score": 0.0
                      },
                      "65": {
                        "Number": 65,
                        "Score": 0.09523809523809523
                      },
                      "66": {
                        "Number": 66,
                        "Score": 0.09523809523809523
                      }
                    }
                  },
                  "System.Void TaskManagers.TaskManager::.ctor()": {
                    "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                    "Lines": {
                      "11": {
                        "Number": 11,
                        "Score": 0.5555555555555556
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.5555555555555556
                      },
                      "14": {
                        "Number": 14,
                        "Score": 0.5555555555555556
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "Simple Matching & Sampaio": {
      "Assemblies": {
        "TaskManagers.dll": {
          "Name": "TaskManagers.dll",
          "Files": {
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\Task.
    cs",
              "Classes": {
                "TaskManagers.Task": {
                  "Name": "TaskManagers.Task",
                  "Methods": {
                    "System.Int32 TaskManagers.Task::get_Id()": {
                      "Signature": "System.Int32 TaskManagers.Task::get_Id()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Title()": {
                      "Signature": "System.String TaskManagers.Task::get_Title()",
                      "Lines": {}
                    },
                    "System.String TaskManagers.Task::get_Description()": {
                      "Signature": "System.String TaskManagers.Task::get_Description
    ()",
                      "Lines": {}
                    },
                    "System.Boolean TaskManagers.Task::get_IsCompleted()": {
```

```
                    "Signature": "System.Boolean TaskManagers.Task::
    get_IsCompleted()",
                    "Lines": {}
                  },
                  "System.String TaskManagers.Task::get_Category()": {
                    "Signature": "System.String TaskManagers.Task::get_Category()"
    ,
                    "Lines": {}
                  },
                  "System.Int32 TaskManagers.Task::get_Priority()": {
                    "Signature": "System.Int32 TaskManagers.Task::get_Priority()",
                    "Lines": {}
                  },
                  "System.Void TaskManagers.Task::Update(System.String,System.
    String,System.String,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::Update(System.
    String,System.String,System.String,System.Int32)",
                    "Lines": {
                      "26": {
                        "Number": 26,
                        "Score": 0.1
                      },
                      "28": {
                        "Number": 28,
                        "Score": 0.1
                      },
                      "30": {
                        "Number": 30,
                        "Score": 0.1
                      },
                      "31": {
                        "Number": 31,
                        "Score": 0.1
                      }
                    }
                  },
                  "System.Void TaskManagers.Task::.ctor(System.Int32,System.String
    ,System.String,System.String,System.Boolean,System.Int32)": {
                    "Signature": "System.Void TaskManagers.Task::.ctor(System.Int3
    2,System.String,System.String,System.String,System.Boolean,System.Int32)",
                    "Lines": {
                      "5": {
                        "Number": 5,
                        "Score": 0.5384615384615384
                      },
                      "7": {
                        "Number": 7,
                        "Score": 0.5384615384615384
                      },
                      "9": {
                        "Number": 9,
                        "Score": 0.5384615384615384
                      },
                      "10": {
                        "Number": 10,
                        "Score": 0.5384615384615384
                      },
                      "11": {
                        "Number": 11,
                        "Score": 0.5384615384615384
                      },
                      "13": {
                        "Number": 13,
                        "Score": 0.5384615384615384
                      },
                      "14": {
                        "Number": 14,
```

```
                        "Score": 0.5384615384615384
                      }
                    }
                  }
                }
              }
            },
            "C:\\Project\\DefectsSampleProject\\TaskManagers\\TaskManager.cs": {
              "SourcePath": "C:\\Project\\DefectsSampleProject\\TaskManagers\\
    TaskManager.cs",
              "Classes": {
                "TaskManagers.TaskManager": {
                  "Name": "TaskManagers.TaskManager",
                  "Methods": {
                    "TaskManagers.Task TaskManagers.TaskManager::AddTask(System.
    String,System.String,System.String,System.Int32)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    AddTask(System.String,System.String,System.String,System.Int32)",
                      "Lines": {
                        "19": {
                          "Number": 19,
                          "Score": 0.5384615384615384
                        },
                        "20": {
                          "Number": 20,
                          "Score": 0.5384615384615384
                        },
                        "21": {
                          "Number": 21,
                          "Score": 0.5384615384615384
                        }
                      }
                    },
                    "System.Boolean TaskManagers.TaskManager::RemoveTask(System.Int3
    2)": {
                      "Signature": "System.Boolean TaskManagers.TaskManager::
    RemoveTask(System.Int32)",
                      "Lines": {
                        "26": {
                          "Number": 26,
                          "Score": 0.0
                        },
                        "27": {
                          "Number": 27,
                          "Score": 0.0
                        },
                        "29": {
                          "Number": 29,
                          "Score": 0.0
                        },
                        "32": {
                          "Number": 32,
                          "Score": 0.0
                        },
                        "33": {
                          "Number": 33,
                          "Score": 0.0
                        }
                      }
                    },
                    "TaskManagers.Task TaskManagers.TaskManager::GetTask(System.Int3
    2)": {
                      "Signature": "TaskManagers.Task TaskManagers.TaskManager::
    GetTask(System.Int32)",
                      "Lines": {
                        "38": {
```

```
                        "Number": 38,
                        "Score": 0.35714285714285715
                      }
                    }
                  },
                  "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetAllTasks()": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetAllTasks()",
                  "Lines": {
                    "43": {
                      "Number": 43,
                      "Score": 0.0
                    }
                  }
                },
                "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByCategory(System.String)": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByCategory(System.String)",
                  "Lines": {
                    "48": {
                      "Number": 48,
                      "Score": 0.1
                    }
                  }
                },
                "System.Collections.Generic.List`1<TaskManagers.Task>
    TaskManagers.TaskManager::GetTasksByPriority(System.Int32)": {
                  "Signature": "System.Collections.Generic.List`1<TaskManagers.
    Task> TaskManagers.TaskManager::GetTasksByPriority(System.Int32)",
                  "Lines": {
                    "54": {
                      "Number": 54,
                      "Score": 0.1
                    }
                  }
                },
                "System.Boolean TaskManagers.TaskManager::UpdateTask(System.Int3
    2,System.String,System.String,System.String,System.Int32)": {
                  "Signature": "System.Boolean TaskManagers.TaskManager::
    UpdateTask(System.Int32,System.String,System.String,System.String,System.Int32)"
    ,
                  "Lines": {
                    "59": {
                      "Number": 59,
                      "Score": 0.18181818181818182
                    },
                    "60": {
                      "Number": 60,
                      "Score": 0.18181818181818182
                    },
                    "62": {
                      "Number": 62,
                      "Score": 0.0
                    },
                    "65": {
                      "Number": 65,
                      "Score": 0.1
                    },
                    "66": {
                      "Number": 66,
                      "Score": 0.1
                    }
                  }
                },
                "System.Void TaskManagers.TaskManager::.ctor()": {
```

```
                        "Signature": "System.Void TaskManagers.TaskManager::.ctor()",
                        "Lines": {
                          "11": {
                            "Number": 11,
                            "Score": 0.625
                          },
                          "13": {
                            "Number": 13,
                            "Score": 0.625
                          },
                          "14": {
                            "Number": 14,
                            "Score": 0.625
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```


Listing F.7: Defects Sample Project - TaskManagers - Suspiciousness Results from Fault Detector .NET CLI