

8-2015

# Formal Specification Driven Development

Titus Fofung  
*Kennesaw State University*

Follow this and additional works at: <http://digitalcommons.kennesaw.edu/etd>

 Part of the [Computer Engineering Commons](#), and the [Software Engineering Commons](#)

---

## Recommended Citation

Fofung, Titus, "Formal Specification Driven Development" (2015). *Dissertations, Theses and Capstone Projects*. Paper 682.

This Thesis is brought to you for free and open access by DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Dissertations, Theses and Capstone Projects by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact [digitalcommons@kennesaw.edu](mailto:digitalcommons@kennesaw.edu).

# Quantitative Analysis of Formal Specification Driven Development

A Thesis Presented to  
The Faculty of the Department of Computer Science and Software Engineering

By

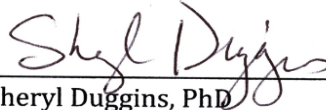
Titus Dohnfon Fofung

In Partial Fulfillment  
Of Requirements for the Degree  
M.S. in Software Engineering

Southern Polytechnic State University  
August 2015

# Quantitative Analysis of Formal Specification Driven Development

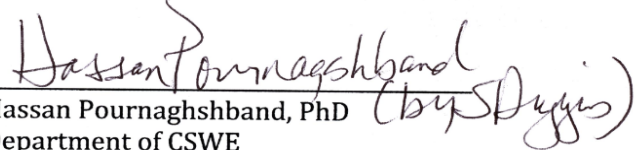
Approved:



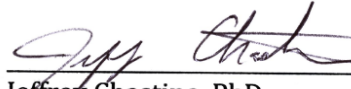
Sheryl Duggins, PhD  
Thesis Advisor



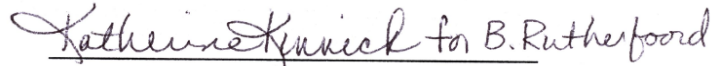
Frank Tsui, PhD  
Department of CSWE



Hassan Pournaghshband, PhD (by Sheryl Duggins)  
Department of CSWE



Jeffrey Chastine, PhD  
Chairman Department of CSWE



E.K. Park, Ph.D. / Becky Rutherford, Ph.D.  
Dean, School of Computing and SWE

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Southern Polytechnic State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis that involves potential financial gain will not be allowed without written permission.



Titus Dohnfon Fofung

Notice To Borrowers

Unpublished theses deposited in the Library of Southern Polytechnic State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

☐The author of this thesis is:

Titus Dohnfon Fofung  
3011 Forbes Trail  
Snellville, GA 30039

The director of this thesis is:

Sheryl Duggins  
SPSU School of CSE, Building J, J365  
1100 South Marietta Pkwy  
Marietta, GA 30060

Users of this thesis not regularly enrolled as students at Southern Polytechnic State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Name	Address	Date	Type of use
------	---------	------	-------------

# Quantitative Analysis of Formal Specification Driven Development

An Abstract of  
A Thesis Presented to  
The Faculty of the Department of Computer Science and Software Engineering

By

Titus Dohnfon Fofung

In Partial Fulfillment  
Of Requirements for the Degree  
M.S. in Software Engineering

Southern Polytechnic State University  
August 2015

This paper researches a quantitative metric of investigating Formal Specification-Driven Development (FSDD). Formal specification is needed at the beginning of the development process to prevent ambiguity and to improve the quality through corrections of errors found in the late phases of a traditional design process, Software Development Life Cycle (SDLC). The research is conducted with capstone students using both the FSDD and the SDLC (traditional) models and a quantitative analysis is presented to evaluate the internal quality of the software. The tool used to measure the internal quality is the .NET 2013 analysis tool.

Formal Specification-Driven Development (FSDD) is a new approach in which formal specification is used and functional units of code are incrementally written and tested prior to the code implementation. In the research, there is a comparative study of Formal Specification-Driven Development with the traditional model. This research realized the promising attributes of Formal Specification Driven Development. It promotes the incorporation of FSDD in the software development process. FSDD is radically different from the traditional ways of developing software. In the traditional software development model (SDLC), the tests are written after code implementation. In FSDD the test occurs during development. This model is more of a programmer's process.

This study is the first complete evaluation of how FSDD affects software development and internal design quality. The study was carried out with students in a Software Engineering Capstone class. The research included a semester-long project to develop a ticketing system.

This research demonstrated that software developers applying Formal Specification-Driven Development (FSDD) approach are likely to improve some software quality aspects compared to Software Development Life Cycle (SDLC) approach. In particular this research has shown significant differences in the areas of code complexity and size statistically. The differences in internal quality can considerably improve the external software quality, software maintainability, software understandability, and software reusability. The research establishes a standard for future studies. It focused mainly on the software process. This research is going to raise awareness of FSDD as a new software development approach to explore further.

# Formal Specification Driven Development

A Thesis Presented to  
The Faculty of the Department of Computer Science and Software Engineering

By

Titus Dohnfon Fofung

Advisor: Sheryl Duggins

In Partial Fulfillment  
Of Requirements for the Degree  
M.S. in Software Engineering

Southern Polytechnic State University  
August 2015



This is dedicated to my sons Gurb Fofung and Bila Fofung and daughter, Nahdia Fofung, for their sustenance, endurance, and consideration throughout this undertaking.

## Table of Contents

<b>1. INTRODUCTION.....</b>	<b>13</b>
1.1. MOTIVATION.....	15
1.2. THE COMPONENTS OF FORMAL SPECIFICATION-DRIVEN DEVELOPMENT (FSDD) 16	
1.2.1. TEST-DRIVEN DEVELOPMENT (TDD).....	16
1.2.2. BEHAVIOR-DRIVEN DEVELOPMENT (BDD).....	17
1.2.3. FORMAL METHODS.....	20
1.3. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC).....	21
1.4. SUMMARY OF THE REMAINING CHAPTERS.....	23
1.5. OBJECTIVE.....	24
<b>2. LITERATURE REVIEW.....</b>	<b>26</b>
2.1. TEST-DRIVEN DEVELOPMENT AND FORMAL METHODS.....	27
2.2. RESEARCH IN TEST-DRIVEN DEVELOPMENT.....	28
2.2.1. TEST-DRIVEN DEVELOPMENT IN ACADEMIA.....	28
2.2.2. TEST-DRIVEN DEVELOPMENT IN INDUSTRY.....	29
2.1. BEHAVIOR-DRIVEN DEVELOPMENT (BDD).....	31
<b>3. RESEARCH METHODOLOGY.....</b>	<b>33</b>
3.1. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT MODEL.....	33
3.1.1. Java Model.....	33
3.2. EXPERIMENTAL DESIGN.....	45
3.2.1. EXPERIMENT OVERVIEW.....	46
<b>4. RESULTS AND ANALYSIS.....</b>	<b>59</b>
4.1. METRICS COLLECTION AND ANALYSIS.....	59
4.2. INTERNAL QUALITY RESULTS.....	59
4.2.1. METHOD-LEVEL METRICS.....	60
4.2.2. CLASS-Level Metrics.....	62
4.2.3. PROJECT-LEVEL METRICS.....	66
4.3. TEST RESULTS.....	70
<b>5. EVALUATION, OBSERVATION, AND DISCUSSION.....</b>	<b>71</b>
5.1. EMPIRICAL EVIDENCE OF FORMAL SPECIFICATION-DRIVEN DEVELOPMENT EFFICACY.....	71
5.1.1. QUANTITATIVE EVIDENCE: COMPLEXITY.....	72
5.1.2. QUANTITATIVE EVIDENCE: COUPLING.....	74
5.1.3. QUANTITATIVE EVIDENCE: COHESION.....	77
5.1.4. QUANTITATIVE EVIDENCE: SIZE.....	79
5.1.5. QUANTITATIVE EVIDENCE: STATIC CODE QUALITY ANALYSIS.....	81
5.1.6. EMPIRICAL EVIDENCE SUMMARY AND CONCLUSIONS.....	82
5.2. SUMMARY AND FUTURE WORK.....	83
<b>6. BIBLIOGRAPHY.....</b>	<b>86</b>
<b>7. APPENDIX A FORMAL SPECIFICATION-DRIVEN DEVELOPMENT APPROACH METRICS.....</b>	<b>92</b>
7.1. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT - ALL METRICS.....	92
7.2. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT CLASS-LEVEL METRICS.....	146
7.3. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT PROJECT-LEVEL METRICS.....	147
<b>8. APPENDIX B SOFTWARE DEVELOPMENT LIFE CYCLE APPROACH METRICS.....</b>	<b>148</b>
8.1. SOFTWARE DEVELOPMENT LIFE CYCLE - ALL METRICS.....	148

8.2.	SOFTWARE DEVELOPMENT LIFE CYCLE METHOD-LEVEL METRICS .....	201
8.3.	SOFTWARE DEVELOPMENT LIFE CYCLE CLASS-LEVEL METRICS .....	205
8.4.	SOFTWARE DEVELOPMENT LIFE CYCLE PROJECT-LEVEL METRICS.....	206
9.	APPENDIX .NET CODE ANALYSIS TOOL.....	207
9.1.	.NET CODE ANALYSIS TOOL.....	207
10.	APPENDIX D METRICS.....	208
10.1.	MAINTAINABILITY INDEX .....	208
10.2.	CYCLOMATIC COMPLEXITY .....	208
10.3.	DEPTH OF INHERITANCE .....	208
10.4.	CLASS COUPLING.....	209
10.5.	LINES OF CODE.....	209

## Table of Figures

Figure 1: Test-Driven Development .....	16
<b>Figure 2: Behavior-Driven Development.....</b>	<b>18</b>
<b>Figure 3: SDLC - Waterfall Model .....</b>	<b>21</b>
<b>Figure 4: The Simplified SDLC Process .....</b>	<b>22</b>
<b>Figure 5: Method-Level Metrics p-values.....</b>	<b>61</b>
Figure 6: % difference in Method-Level Metrics.....	61
<b>Figure 10: Class-Level Metrics p-values .....</b>	<b>63</b>
Figure 11: % Difference in Class-Level Metrics.....	64
Figure 12: Box plot for line of code in Classes .....	64
Figure 12: Box plot Class-Level Maintainability Index.....	65
Figure 14: % Difference Project-level Metrics.....	67
Figure 16: Difference in Project-Level Cyclomatic Complexity.....	73
Figure 17: Difference in Depth of Inheritance .....	73
Figure 17: Difference in Project-Level Cyclomatic Complexity.....	74
Figure 19: Box plot of Method-Level Class Coupling.....	76
<b>Figure 20: Box plot of Class-Level Class Coupling .....</b>	<b>77</b>
<b>Figure 17: Differences in Cohesion Metric CC .....</b>	<b>78</b>
<b>Figure 18: % Difference in Size Metrics.....</b>	<b>79</b>
<b>Figure 18: Static Code Quality.....</b>	<b>81</b>

## List of Tables

Table 1: C# Code Metrics .....	52
Table 2: Sample Metrics by attribute .....	53
Table 3: Summary of Methods Metrics .....	60
Table 4: Summary of Class-level Metrics .....	62
Table 5: Project-Level Metrics results .....	66
Table 6: FSDD Project-level Static Code Analysis results .....	68
Table 7: SDLC Project-level Static Code Analysis results .....	69
Table 8: Project-Level Metrics results .....	82
Table 9: FSDD Metrics .....	92
Table 10: FSDD Class-Level Metrics .....	146
Table 11: FSDD Project-Level Metrics .....	147
Table 12: SDLC Metrics .....	148
Table 13: SDLC Method-Level Metrics .....	201
Table 14: SDLC Class-Level Metrics .....	205
Table 15: SDLC Project-Level Metrics .....	206

# 1. INTRODUCTION

The study is aimed at providing an effectiveness comparison between the quality of Formal Specification-Driven Development (FSDD), and the Software Development Life Cycle (SDLC) methods of creating software. Unlike SDLC, FSDD is a new approach in software development proposed by Rutledge & Tsui (2013), in which units of programming codes are incrementally written and tested prior to system implementation. The formal specification is needed at the beginning of the development process to prevent ambiguity and to improve the quality of software. FSDD has been improved through corrections found in the late phases of a traditional design process (Rutledge & Tsui, 2013). With the traditional software development model, the tests are written after the code implementation. The SDLC test is mainly for implementation and not development, and hence referred to as “test-last”. Meanwhile, with the FSDD model, the test occurs during development, and is more of a programmer’s process in which testing is done during the coding phase by the developer. FSDD is thus classified as “test-first” since it involves the use of Test-Driven Development (Erdogmus et al. 2005).

This study will be conducted with two groups of capstone students from Southern Polytechnic State University, who will respectively develop a system using FSDD and SDLC models to provide the data for the comparative quantitative analysis. This quantitative analysis will help to conclude on the software effectiveness and quality.

Software failure occurs very frequently and so there is a need to remedy the situation. The introduction of the Formal Specification Driven Development proposed by Richard Rutledge in his 2013 thesis needs to be investigated for its effectiveness, as compared to other methodologies, for possible incorporation into software development. Test-Driven Development (TDD) and Behavior-Driven Development (BDD) using formal specification are the basis of FSDD (Rutledge & Tsui, 2013). FSDD could be classified as an agile software development approach derived from Extreme Programming (XP) (Beck, 2001). While FSDD is new, TDD and BDD have been around for more than a decade (Beck, 2002). Software engineers always have to look for new ways to improve software quality. FSDD can be categorized as formalized TDD (Rutledge & Tsui 2013). This thesis will investigate the new process and present an argument that FSDD comes with the benefits that TDD and BDD have, but without their shortcomings (Tsui, 2010). The study will attempt to establish the quantitative efficacy of FSDD. The various aspects involved in creating a balanced condition will be discussed and determined. The other component of FSDD, the TDD, was introduced in 2003 but is not extensively used as much (Rutledge, & Tsui 2013). Also, FSDD includes BDD and Formal Methods in the development approach.

The study had intended to get test data as direct evidence but resorted to the use of quality metrics since only one team submitted their test data. This was due to some logistic constraints. The SDLC team could not be readily contacted after their code was submitted. It then became apparent that the Visual Studio Analysis tool was the only option to work with. There were three teams at the beginning but could only utilize the code from the teams that used Visual Studio, hence the one team was dropped since they used PHP to code their project.

## 1.1. MOTIVATION

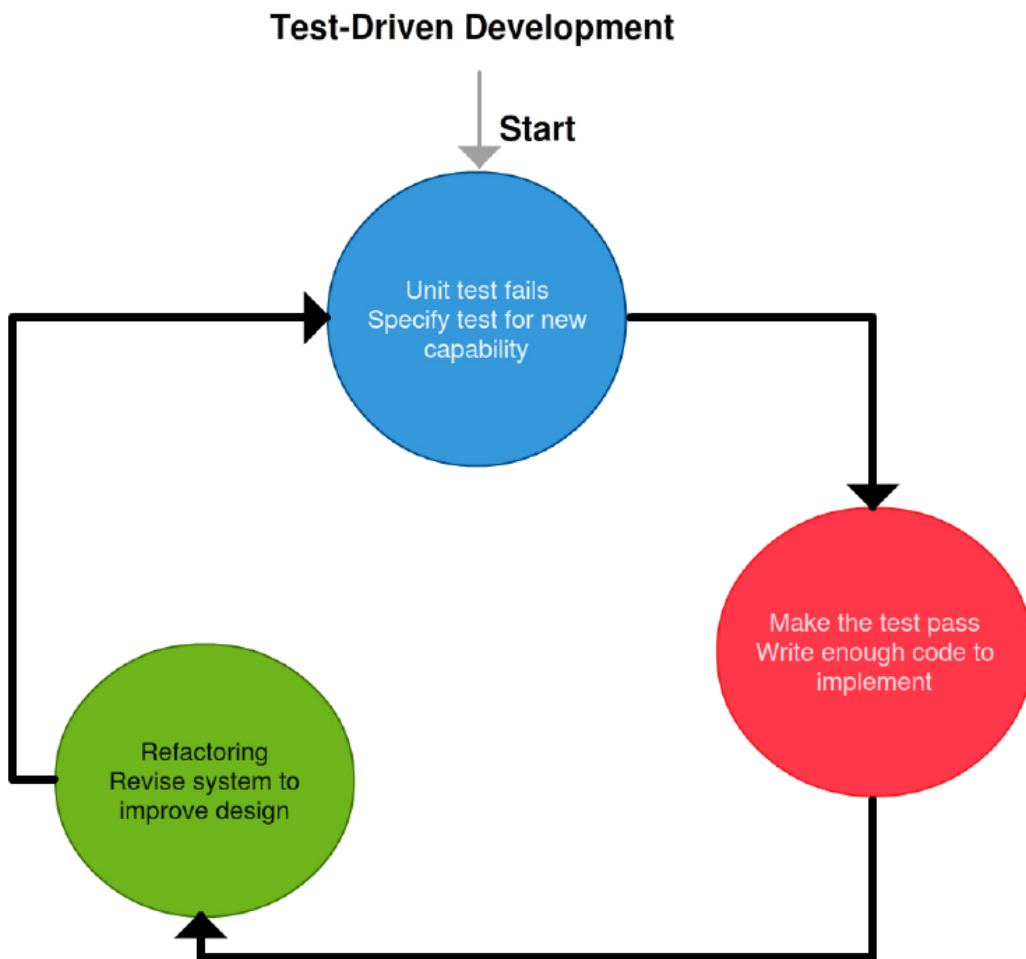
The proliferation of computer software usage around the world has reached an enormous proportion, such that software development has become more and more complex, leading to increasingly compromised quality. Dependence on computer software for day-to-day living may not be very apparent. Software is involved in healthcare, agriculture, transportation, communication, and leisure, just to name a few areas, and has become inevitable in today's lifestyle worldwide (Preserve Articles, 2011). The correct functioning of the software is of utmost importance, especially in critical systems that cannot afford to fail or be defective. Therefore, software engineering should strive for 100% accuracy, although this seems utopian. But the introduction of the various development methods such as Test-Driven Development and Behavior-Driven Development, and now Formal Specification-Driven Development, are part of the ongoing strategies that attempt to reduce the incidence of software defects and failures. These methodologies may stem the tide of software failures and defects. This study is a further investigation of the work started by Rutledge (2013). This technique proves to be a potentially useful tool in a reduction of software failure. But this cannot be concluded unless a comparison to existing methodologies is achieved. The prospects of the FSDD approach have motivated me to further investigate its usefulness.



## 1.2. THE COMPONENTS OF FORMAL SPECIFICATION-DRIVEN DEVELOPMENT (FSDD)

### 1.2.1. TEST-DRIVEN DEVELOPMENT (TDD)

TDD is the exercise of writing each piece of production code in direct response to a test. But if the test fails, the production code is rewritten until the test passes. This definition is only a simplified description of a very complex process. In TDD, the tests are written in a stepwise process with each step written to completion with passing code before the next is started (Kumar & Bansal, 2013).

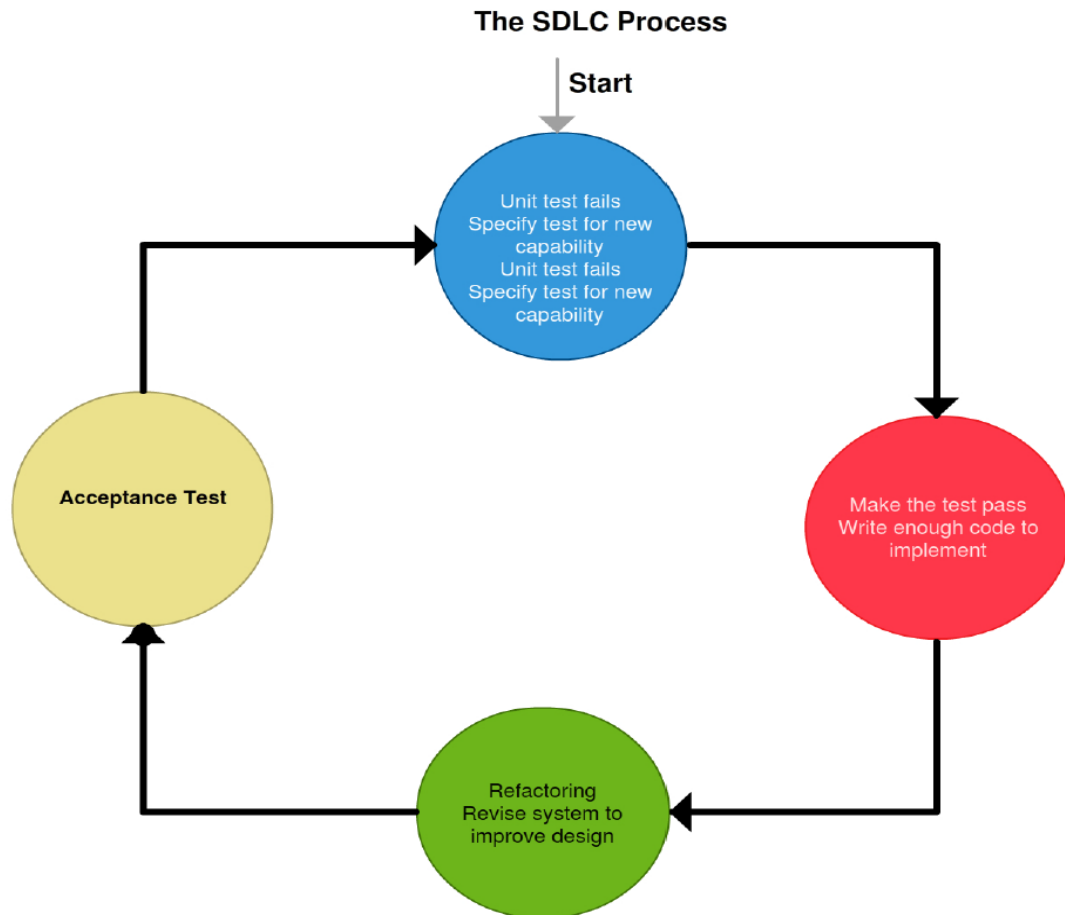


**Figure 1: Test-Driven Development**

Generally, this means that in TDD just a small part of the test is written at a time, and it begins with a simple, uncomplicated example, and the code is tested to make it work, typically within a few minutes, before proceeding to the next test portion (Erdogmus, et al. 2005) to meet the requirements and constraints that the tests provide. But as the tests get more and more challenging, the code becomes more and more capable. Summarily, TDD is more concerned with the testing of a component as a unit (Shull, et al. 2014). The testing is free of the other dependencies, but it does not entirely replace the usual conventional software testing. It increases software quality by improving correctness.

#### **1.2.2. BEHAVIOR-DRIVEN DEVELOPMENT (BDD)**

BDD focuses on the users' opinion of how the software should behave. It is more about functionality and so must start with the most important function pertinent to the users of the system. When the crucial function has been identified, the developer then takes over and implements it. BDD deals with business domain, and it is used for acceptance and regression testing (Solis & Xiaofeng, 2011). Figure 2 is a depiction of BDD.



**Figure 2: Behavior-Driven Development**

BDD is mostly regarded as the evolution of TDD. BDD is centered on defining very precise specifications. BDD is specified with the Gherkin Language. Gherkin defines example scenarios in a Given-When-Then format to create structure around behaviors so that they can be automated. Gherkin uses a simple structure for documenting examples of the behavior the stakeholders want, bridging the communication between developers and stakeholders (Wynne & Hellesoy, 2010). In conversations to promote a shared understanding, Gherkin uses user-defined tags to organize scenarios.

BDD focuses on behavioral aspects while TDD emphasizes implementation. In BDD, the system is described in a way that can be easily automated. BDD provides a precise,

uncomplicated language that helps stakeholders to specify their tests (Solis & Xiaofeng, 2011). The transparency between user expectation and developers' tests is a significant advantage. The toolkits supporting BDD include Cucumber (JUnit.org, 2004) and RSpec (Beck & Gamma, 1998). RSpec is a BDD framework for the Ruby programming language. Cucumber is a software tool used for testing other software. Cucumber is created using the Ruby programming language. Gherkin only has a few keywords that enable the building of a domain specific language for everyday use in the system.

The BDD approach is still in its infancy, just like FSDD, so it could be characterized as being under development. The BDD concept is still a little vague. There is no one commonly accepted definition of BDD. The characteristics of BDD are not concise. The tools are mainly concentrated on the implementation phase of the development process, which is incompatible with BDD's wider involvement in the software development lifecycle.

BDD could be explained further with an example. Consider sorting with various types of method as thus:

- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Comb sort
- Merge-sort

- Quicksort

The test for sorting in BDD could be written using the Cucumber tool as follows:

- ✓ **Given a list of numbers**
- ✓ **When I sort the list**
- ✓ **Then the list will be in numerical order**

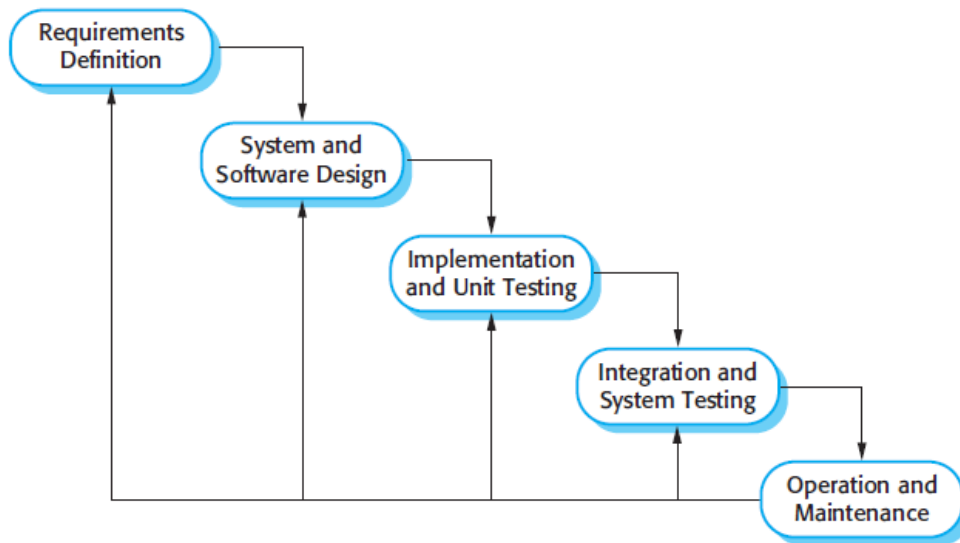
The sorting method does not matter, or the routine employed to test and implement the sort; all that is needed is a sorted list (Falco, 2013).

### **1.2.3. FORMAL METHODS**

Formal methods (FM) are another component of FSDD which are mathematical ways of modeling a system. They are scientifically based and hence are considered to be very reliable and accurate (Staples, 1996). They help in avoiding ambiguities in the specification of software. FMs are, unfortunately, not very welcomed by programmers, due to their mathematical aspects, since many people have very little affinity for mathematics. FMs are rarely used in the industry on a large scale, though it plays a significant role in reducing software defects (Boehm & Basili, 2001). There are many types of formal methods, including Vienna Development Method (VDM), Object Constraint Language (OCL), Z and Java Modelling Language (JML). FMs will eventually pay off if learned and used in the software development process. When the specifications are correct, then less time will be spent on software testing. Using FMs is a great approach for building reliable software (Wedde et al. 1992).

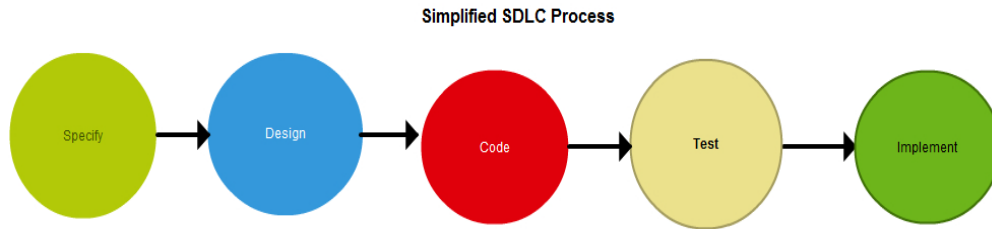
### 1.3. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

There exist many different types of traditional software development techniques, with the most common being the Waterfall Model. With the traditional methods, all tests are carried out after the code has been implemented (Janzen & Saiedian, 2008). The distinct phases of the Waterfall Model are as illustrated in Figure 2 (Sommerville, 2010, p. 30).



**Figure 3: SDLC - Waterfall Model**

The Waterfall Model is a plan-driven development. For this model, documentation is very critical, and so a written plan and schedule of all the process activities must be done before embarking on the software development (Sommerville, 2010). Figure 3 summarizes the process as: Specify, Design, Code, Test and Implement.



**Figure 4: The Simplified SDLC Process**

This study will demonstrate why there is a need to investigate the new processes of software development. It is imperative to know how FSDD fares against SDLC. It will be important to examine the usefulness of FSDD compared to the traditional methods for potential application in software development in the future. Very little research has been conducted using FSDD compared to the vast amount of research data available for TDD and a little for BDD. While it is a relatively new technique, a statistical study for FSDD is needed for quality and productivity. Therefore, this study will concentrate on the quality deficiency, with the purpose of determining if this formalized combination of the TDD and BDD approaches is a better technique in curbing the software defects that are regularly encountered with using the traditional method of development. The FSDD technique does not eliminate testing at the end of the software development, but it reduces the time devoted to the system testing phase. The significance of all of these will usher in a new way of developing systems, especially critical systems. Analysis of the collected data will help reveal the accuracy of FSDD compared to SDLC, and potentially provide a guide to whether FSDD is a viable software engineering method of the future. The research will show that FSDD is a sustainable software development technique of the future.

In this study, analysis of the data collected will help portray the accuracy of FSDD compared to SDLC, and potentially provide a guide to whether FSDD is a practical and the prospective software engineering method. In other words, it will be important to explore the usefulness of FSDD compared to the traditional methods for potential application of FSDD in software development in the future.

#### **1.4. SUMMARY OF THE REMAINING CHAPTERS**

Chapter 2 will provide a literature review of studies on FSDD (including FM, TDD & BDD) techniques. The development processes will be discussed, with historical references to some evolving forms of FSDD. References to TDD, BDD, and FM will be noted, since FSDD is still a very new approach. Recent literature in which FM, TDD, and BDD have gained some prominence will be considered.

Chapter 3 will describe the research methodology that will be utilized in the study. It will explain in detail how the research is carried out. The chapter starts with an example of how FSDD is used in software development. It will attempt to provide a step-by-step process, with an example to make the FSDD technique very comprehensible. The tools and metrics utilized are also identified, the objectives and hypotheses are discussed, and the methods and data acquisition are highlighted.



Chapter 4 gives the results of the research conducted. The data is analyzed and discussed. Finally, Chapter 5 summarizes the work and debates its potential to improve the quality of software, and stresses its importance. Future work will be identified.

### **1.5. OBJECTIVE**

This study is a follow-up of the qualitative study carried out by Richard Rutledge, who introduced the FSDD technique. Therefore, this quantitative aspect of the study is done to evaluate the effects of FSDD on software quality when compared to the traditional SDLC approach. The samples shall be collected from two groups of software engineering students working on their capstone projects. The independent variable is the use of FSDD versus SDLC development. The dependent variable is software quality. These two sets of independent samples will be analyzed to answer the research question that is mentioned below. Meanwhile, in doing this analysis, to control for confounding by the independent variables, the covariates that include the experience and programming skills of the students and the number of them involved shall be controlled.

#### **1.5.1 Research Question (RQ) and Hypothesis (HP<sub>0</sub>)**

RQ: Is the internal quality of software developed using Formal Specification-Driven Development (FSDD) technique higher than that using the traditional approach (SDLC)?

Null Hypothesis (HP<sub>0</sub>) When the Formal Speculation-Driven Development (FSDD) and the (SDLC) techniques are compared, there is difference found in the internal quality of the software.

**$\text{IntQlty}_{FSDD} = \text{IntQlty}_{SDLC} \implies \text{p-value} > 0.0500$**

Alternative Hypothesis: When the (FSDD) and the (SDLC) techniques are compared, a significant improvement in quality of the software developed is noted.

**$\text{IntQlty}_{FSDD} > \text{IntQlty}_{SDLC} \implies \text{p-value} < 0.0500$**

## 2. LITERATURE REVIEW

This chapter summarizes and evaluates the studies related to TDD, BDD, FM, and SDLC. There are some publications on XP and Agile methods, many anecdotal and some empirical. The first section will present some observations that validate the notion that TDD can be studied and applied independently of XP. However, this discussion will exclude research on XP or Agile methods as a whole. Such research on agile methods might prove informative when examining TDD, but it fails to show any individual merits or shortcomings of TDD.

After a thorough search of three relevant electronic databases (IEEE Explorer, ACM Digital Library, Georgia Library Learning Online - GALILEO), no research article was found that compares FSDD to the traditional technique. These databases were selected based on the fact that they cover most of the relevant conferences and peer-reviewed publications, and were easy to access. In the literature search, the keywords: Traditional Method, Test-Driven Development, TDD, Behavior-Driven Development, BDD, Formal Specification-Driven Development, FSDD, and Formal Methods were used.

More than seven searches were done for each of the sources, using different permutations of some keywords. The searches included BDD and Traditional Method; TDD and Traditional Method; Comparative Study of Formal TDD and Traditional Method; and Formal Methods and TDD. The search results were manually checked based on titles and abstracts. The following articles were found in the search.

## **2.1. TEST-DRIVEN DEVELOPMENT AND FORMAL METHODS**

Aichernig, Lorber & Tiran (2012) provide information about formal refinement and Test-Driven Development used in the study of car alarm systems. Formal methods are introduced to generate test cases. This study is similar to the concept of FSDD. However, it does not compare this process with the traditional technique.

Baumeister (2004) showed that combining formal specifications with TDD does not necessarily lead to losing the agility of TDD, and also proposes the formalization of TDD using JUnit tests instrumented with run-time assertions generated from the JML specifications. The concept was similar to that of FSDD but did not include a comparative study.

Alawneh & Peters' (2013) paper discusses the use of TDD, formal specifications and the right tools to develop programs. This study is very much in concert with FSDD. The paper proposes the use of formal notation to specify the behavior as an alternative to TDD. The research talks about FSDD without naming it as such. The new name was not given to this proposed method; it could have been, since it does not deviate that much from FSDD concept.

Beck (2001) argues that TDD is a code development process that has been made very famous by extreme programming, but it is not a testing technique. Rutledge & Tsui (2013) propose the FSDD technique of software development, pulling the knowledge from the improvements that have been made with TDD and BDD to decrease errors and increase the quality.

## **2.2. RESEARCH IN TEST-DRIVEN DEVELOPMENT**

Research on TDD can be categorized broadly by context. In particular, TDD research will be classified as “industry” if the study or research was primarily conducted with professional software practitioners. Alternatively, the research will be classified as “academia” if the software developers are mainly students and the work is in the context of a course or some other academic setting.

### **2.2.1. TEST-DRIVEN DEVELOPMENT IN ACADEMIA**

A paper by Erdogmus et al. (2005) shows that TDD offered no change in software quality, but there was an improvement in productivity when the study was conducted with undergraduate students. In Fucci & Turhan's (2013) research, the same controlled experiment by Erdogmus et al. was carried out and concluded that there was no noteworthy change in productivity and quality. Edwards (2003) also conducted a study of TDD and found a significant increase in quality and productivity by way of project assignments to undergraduate students.

Kaufmann, & Janzen (2003), realized an increase in quality and programmer productivity with 8 students in their study, similar to that of Edwards, 2003, who carried out his study with 59 students. Muller & Hagner (2003) found no significant difference in quality and productivity, but better reuse in their study, using 19 students. Pancur et al. (2003) also realized no change in quality and programmer productivity in their study, using 38 students.

Most of the literature reviews were more focused on the academic studies, since they were similar to the way this research will be conducted. These studies were done using TDD without formal methods, except Aichernig, Lorber, & Tiran (2012), Baumeister (2004) and Alawneh & Peters (2013).

### **2.2.2. TEST-DRIVEN DEVELOPMENT IN INDUSTRY**

There have been some attempts to study software quality and developer productivity using TDD in industrial settings. George and Williams (2004) carried out their study with 24 professional pairs of programmers in three companies. In the development of a bowling application, the pairs were selected at random to a TDD or an SDLC group, and the projects assessed at completion. They determined TDD produced superior external code quality using a set of blackbox test cases. The TDD passed 18% more test cases than the SDLC, but the TDD group spent 16% more time on this approach. The post-development interview showed 78% of the subjects favored TDD to improve programmers' productivity.

In Maximilien and Williams (2003), using IBM Retail Store Solutions with a TDD approach and unit testing, the software quality defect fell by 50%, as measured by Functional Verification Tests (FVT). The development time was not impacted, since the process was on time.

Williams et al.'s (2003) case study in IBM reviewed TDD again. Compared to Maximilien and Williams' (2003) project developed with SDLC, defects reduced by 40% using TDD, with regression tests. There was no change in the productivity. Geras et al. (2004) secluded TDD from other XP practices and investigated the effect of TDD on developer productivity and software quality. In their study, TDD does not require more time, though developers in the TDD group wrote and executed more tests. The reason for the shorter time is because less time was spent on debugging the code.

Another study of TDD conducted at Microsoft (Bhat and Nagappan, 2006) reported significant improvement in software quality. There were two projects considered, namely, project A and project B. Using TDD with project A reduced defect rate by 2.6 times and TDD with project B reduced it by 4.2 times as compared to the organizational average. The time factor was as follows: 35% more development time in project A, and 15% more development time in project B, as compared to time spent in non-TDD projects.

Damm and Lundberg (2006) took 1.5 years to conduct longitudinal case studies with 100 professionals at Ericsson. It involved the development of mobile applications using C++ and Java. This study was found to have reduced the project cost by 5-6%, the defect by 5-30% and defect cost by 55%. Sanchez et al.'s (2007) was an extended five-year single case study at IBM, which included 9-17 developers working on a device driver. During this study it was noted that TDD introduction showed an increase of 19% in development time and a 40% increase in internal defect rate.

Janzen and Saiedian's (2008) collection of three industrial experiments and one case study was composed of intersecting teams and individuals. The studies produced some

moderate results favoring TDD considering test coverage and some size metrics, but were inconsistent for complexity, coupling and cohesion measures. The study involved real-world J2EE applications ranging from 800 to 50,000 lines of code. Test coverage was improved with the use of TDD in all the studies except one.

## **2.1. BEHAVIOR-DRIVEN DEVELOPMENT (BDD)**

There have been very few publications about BDD. Among the few, there have been attempts to treat BDD as an evolved form of TDD. Carvalho et al. (2008) characterize BDD as a specification technique. But Tavares et al.'s emphasis is on BDD as a design technique, with a combination of verification and validation in the design phase. As BDD is firmly grounded on the automation of tasks and tests, they advocate proper tooling to support it.

Tavares et al. (2010) also emphasize BDD as a design technique. They, like Carvalho et al., claim BDD is to bring together verification and validation in the design phase. It means thinking of client criteria before embarking on the design of the discrete part that makes up the functionality. They believe in automation of specification and tests, and tools to support these processes.

Keogh (2010) put forward a wider understanding of BDD, and disputes its significance to the entire lifecycle of software development, specifically to the business side and the collaboration between business and software development. He talks about writing a BDD starting with events and their outcomes. He also disputes that BDD defines behavior. Even though Keogh does not provide a complete list of BDD characteristics, he shows in a compelling way that BDD has a wider consequence to software development processes and is not merely a form of TDD.



Lazar et al. (2010) discussed BDD as an important aspect of the business domain and the interaction between business and software development. They said BDD allows domain experts and software developers to communicate seamlessly. In the BDD process, communication of the business and technology worlds refer to a particular system the same way. Any system should have a recognized, confirmable value to the business. Their approach does not take into consideration the rapport among other BDD concepts, like an iterative breakdown process.

## 3. RESEARCH METHODOLOGY

This chapter presents the FSDD approach and details how this study will look at it. FSDD will be introduced in the first section with a small sample application, giving the example in Java.

### 3.1. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT MODEL

#### 3.1.1. Java Model

A Java example of the bank account system will be discussed. JUnit is the testing framework for Java, and the model will use JUnit and FSDD to develop this application. Using FSDD, the specification is written in JML. Here is a case of a simple bank account specification in JML.

##### 3.1.1.1. NATURAL LANGUAGE SPECIFICATION

The description below is a natural language or an informal specification of a simple bank account.

- An account must contain a certain amount of money (*balance*) and is associated with a minimum number that this account may have (*the minimum balance*).
- It is possible to deposit or withdraw an account. A withdraw operation is only possible if there is enough money in the account.
- One or several last deposit or withdraw operations may be canceled.
- The lowest balance of the account may be altered.

### 3.1.1.2. JML SPECIFICATION

The class Account and the class History are implemented in Java, using JML specification. The specifications are not very detailed, in order to keep it simple. To prevent attributes from modifications, they are declared private, and access methods are defined: getBalance, getMin and getHistory in class Account and getBalance and getPrec in class History. Since there are no changes in the methods, they are specified explicitly in JML. The class Account is invariant because the balance of the account must always exceed the minimum balance.

```
/* Class of bank accounts. */
```

```
public class Account {
```

```
    /* Invariant of class Account. */
```

```
    /*@ public invariant getBalance() >= getMin(); */
```

```
    private int balance; // Account balance
```

```
    private int min; // Account minimum balance
```

```
    private History hist; // List of account history
```

```
    /* The balance of this account. */
```

```
    public /*@ pure */ int getBalance() {
```

```
        return balance;
```

```

}

/* The history list of the account. */

public /*@ pure */ History getHistory() {

    return hist;

}

/* The minimum balance of this account. */

public /*@ pure */ int getMin()

    { return min;

}

```

The constructor of class BankAccount constructs an account with a specified balance and the specified minimum balance. Its pre-condition asserts that the specified balance is more than the specified minimum balance.

```

/* Constructs an account with a balance and a minimum balance. */

/*@ obliges that the balance >= min; */

public BankAccount (int balance, int min) {

    this.balance = balance;

    this.min = min;

```

```

        this.hist = null;

    }

```

Since the minimum balance *min* is private, we have to use a method *setMin* (int min) to modify its value; it could be set to any specified value. Its pre-condition asserts that the balance is greater than the minimum value.

```

    /* Sets the minimum balance to a specified value. */

    /*@ pre-condition is getBalance ( ) >= min; */

    public void setMinimum (int min) {

        this.min = min;

    }

```

The method *Deposit* (int amount) deposits the account with the specified amount. The pre-condition obliges the amount to be positive. Its post-condition asserts that the new balance is the former balance increased by the amount, a new history is created with balance from the previous balance of the account, and, with prior history, the history of the account. Its exceptional post-condition asserts that the method should not terminate abruptly.

```

    /* Deposits this amount into the account. */

    /*@ requires amount >= 0;

    /*@ ensures getBalance ( ) == \old (getBalance ( )) + amount &&

```

```

*@ \fresh (getHistory ( )) &&

*@ getHistory ( ).getBalance ( ) == \old (getBalance ( )) &&

*@ getHistory ( ).getPrec ( ) == \old (getHistory ( ));

*@ signals (Exception e) false;

*/

public void deposit(int amount) {

        hist = new History (balance, getHistory ( ));

        balance = balance + amount;

}

```

The *Withdraw* operation is similar to that of the *Deposit*, in addition to the precondition that the balance decreases by the specified amount is more than the minimum balance. The method *Cancel* eliminates the last deposit or debit operation. Its pre-condition stipulates the history to not be null. There must be at least one operation of *Deposit* or *Withdraw* since the account was created. Its post-condition guarantees that the balance and the history have been accommodated in the account with their former values. Below is a sample JML.

```

/* Cancels the last deposit or debit operation. */

/*@ requires getHistory ( ) != null;

```

```
*@ ensures getHistory ( ) == \old (getHistory ( ).getPrec ( )) &&  
  
*@ getBalance ( ) == \old (getHistory ( ).getBalance ( ));  
  
*@ signals (Exception e) false;  
  
*/  
  
public void cancel ( ) {  
  
    balance = hist.getBalance ( );  
  
    hist = hist.getPrec ( );  
  
    }  
  
} // End of class BankAccount
```

### **3.1.1.3. WRITING THE CODE (CREATING THE TEST PROJECT IN ECLIPSE)**

- The Java project is created using eclipse
- The junit.jar is added to the build path of the project
- Two directories are created; one called src for source code and test for test code

#### **3.1.1.3.1. CREATING THE TEST CLASS**

- A new class is added to the test project giving it a recognizable name to signify that the class will be tested.
- `JUnit.framework.TestCase` is imported and has the class extend it.

```
package talkuml.fsdd.examples.banking;  
import junit.framework.TestCase;  
public class BankAccountTests extends TestCase {  
}  
}
```

### 3.1.1.3.2. CREATING TEST METHOD

- The method is declared public with a name that starts with “test”, followed by the description of the test. The rules for creating the methods are as follows:
- All test methods must be declared public and start with the test because JUnit uses reflection to find, recognize and execute the method. This shows how the test methods are identified.
- Test methods must not use, pass parameters or return values

```
package talkuml.fsdd.examples.banking;  
import junit.framework.TestCase;  
public class BankAccountTests extends TestCase {  
    public void testWithdrawWithSufficientFunds () {  
    }  
}  
}
```



### 3.1.1.3.3. WRITING THE TEST ASSERTION

- The code needed to execute the test is not written unless the required assertions code is satisfactory.
- The assertions are worked on, one at a time.

```
public void testWithdrawWithSufficientFunds(){  
    assertTrue(account.getBalance() == oldBalance - amt);  
}
```

The test code above will confirm that when we have sufficient funds in the bank account, after the withdrawal, the balance is the previous balance minus the amount withdrawn.

### 3.1.1.3.4. WRITING THE TEST FRAME

The test code necessary to execute the scenario is written. In this scenario, there is an initial balance of \$600 in the bank account and a \$350 withdrawal as shown below.

```
public void testWithdrawWithSufficientFunds(){  
    BankAccount account = new BankAccount (600);  
    float amt = 350;  
    float oldBalance = account.getBalance();  
    account.withdraw(amt);  
    assertTrue(account.getBalance() == oldBalance - amt);  
}
```

### 3.1.1.3.5. WRITE THE CODE TO PASS THE TEST

Below is simple code needed to pass the test. This example is a simple Java code and is straightforward to comprehend.

```
public class BankAccount {  
    private float balance = 0;  
    public BankAccount(float initialBalance){  
        balance = initialBalance;  
    }  
    public void withdraw(float amt){  
        balance = balance - amt;  
    }  
    public float getBalance() {  
        return balance;  
    }  
}
```

### 3.1.1.3.6. NEXT TEST

This test involves the Deposit method that is similar to the Withdraw method. The test code above will confirm when a deposit is made to the bank account. After a deposit, the balance is the previous balance plus the amount withdrawn.

```
public void testDepositAmountGreaterThanZero(){  
    BankAccount account = new BankAccount (600);  
    float amt = 350;  
    float oldBalance = account.getBalance();  
    account.deposit(amt);  
    assertTrue(account.getBalance() == oldBalance + amt);  
}
```

```
public void testDepositAmountGreaterThanZero(){  
    BankAccount account = new BankAccount (600);  
    float amt = 350;  
    float oldBalance = account.getBalance();  
    account.deposit(amt);  
    assertTrue(account.getBalance() == oldBalance + amt);  
}
```

```
public class BankAccount {  
    private float balance = 0;  
    public BankAccount (float initialBalance){  
        balance = initialBalance;  
    }  
    public void withdraw(float amt){  
        balance = balance - amt;  
    }  
    public float getBalance() {
```

```

        return balance;
    }
    public void deposit(float amt) {
        balance = balance + amt;
    }
}

```

### 3.1.1.3.7. CODE DUPLICATION REMOVED

```

public class BankAccountTests extends TestCase {
    private BankAccount account;
    private float amt;
    private float oldBalance;
    public void setUp(){
        account = new BankAccount (600);
        amt = 350;
        oldBalance = account.getBalance();
    }
    public void testWithdrawWithSufficientFunds(){
        account.withdraw(amt);
        assertTrue(account.getBalance() == oldBalance - amt);
    }
    public void testDepositAmountGreaterThanZero(){
        account.deposit(amt);
        assertTrue(account.getBalance() == oldBalance + amt);
    }
}

```

```

    }
}

public class BankAccountTests extends TestCase {
    private BankAccount account;
    private float amt;
    private float oldBalance;
    public void setUp() {
        account = new BankAccount (600);
        amt = 350;
        oldBalance = account.getBalance();
    }
    public void testWithdrawWithSufficientFunds(){
        account.withdraw(amt);
        assertTrue(account.getBalance() == oldBalance - amt);
    }
    public void testDepositAmountGreaterThanZero() {
        account.deposit(amt);
        assertTrue(account.getBalance() == oldBalance + amt);
    }
}

```

### 3.1.1.3.8. START THE PROCESS AGAIN

The process is repeated here below:

```

public void testWithdrawWithSufficientFunds() {

```

```

    try {
        account.withdraw(amt);
    } catch (AccountException e) {
        fail();
    }
    assertTrue(account.getBalance() == oldBalance - amt);
}

public void testWithdrawWithInsufficientFunds(){
    try {
        account.withdraw(1000);
        fail(); // at this point the test failed
    } catch (AccountException e) {
        // at this point a correct exception was thrown
    }
}

public void withdraw(float amt) throws AccountException {
    if(amt > balance) {
        throw new AccountException();
    }
    balance = balance - amt;
}

```

### 3.2. EXPERIMENTAL DESIGN

This section will outline the details of the formal experiment. It will discuss the hypothesis, independent and dependent variables, the software development process context, and the methods of making and analyzing observations. The method used to analyze the experiment data and how the results were weighed and validated will be discussed. Actual research results will be given in the next chapter.

### **3.2.1. EXPERIMENT OVERVIEW**

This section will describe the study conducted in a capstone class in more detail. The research design will be discussed, including specific artifacts collected, information on the FSDD training provided to students, and descriptions of the projects completed by the students.

The experiment was designed for in a capstone class at Southern Polytechnic State University (SPSU) to collect artifacts. This experiment took place in a capstone class with graduate and undergraduate students at Southern Polytechnic State University Georgia in the software engineering program. The capstone class project involves the design and implementation of software and regularly includes a semester-long team-based project.

The course includes both undergraduate seniors and graduates with different academic but similar professional experiences in their computing background. All students in the course had diverse educational backgrounds. The course met two evenings a week for a sixteen-week semester.

The capstone students were split into three groups; each group had four students. The three groups were given a project to design and implement a ticketing system during the semester. This project was focused on the process of listing event tickets to be sold, customers viewing those events, purchase of tickets, management of tickets to events, and the addition of new events and tickets. Most of the students were familiar with Formal Methods and must have taken the course prior to the capstone course, since it is part of the software engineering curriculum at SPSU.

All the students, including the undergraduate seniors and graduate students, were avid programmers or had taken courses to be able to write some good code. There were four graduate students in the capstone class, who formed one group. This group was charged with the use of the Formal Specification-Driven Development technique, and the other two control groups, undergraduate groups, used the traditional software development approach (SDLC). The FSDD team, on the other hand, used the new method, to allow the detailed design to emerge as the software was developed. The FSDD team was asked to document their detailed design after the code was developed. The graduate group (FSDD team) and one undergraduate team (SDLC team) used Visual Studio 2013 and coded with ASP and C#. One of the undergraduate teams used PHP for their project and hence their code was not used for the comparative studies. They all presented their finished projects on the last day of class and handed in their code.



### **3.2.1.1. FSDD EDUCATIONAL MATERIALS**

The FSDD team was provided with a one-hour guest lecture early in the semester. The talk covered some Java fundamentals as well as training on JUnit and FSDD.

The FSDD team was also offered additional assistance to students struggling with FSDD technique. On a few occasions, the FSDD students requested minor help through email and electronic meetings. FSDD examples were also furnished to the FSDD team.

All teams completed a software requirements specification and high-level architectural design. Educational materials were developed and given to the FSDD team. Information on the FSDD training was provided to the graduate students in the team that made up the observed group. The study provided information on the FSDD approach on how to use Formal Methods and automate unit testing. FSDD information and training were offered only to the FSDD group. The training used some examples of how the technique could be realized.

### **3.2.1.2. TEAM AND TREATMENT SELECTION**

In the software engineering courses, students worked in teams of four programmers most of the time. There were only four graduate students in the capstone class, so they were asked to form a team, and the remaining eight undergraduates were divided into two teams in which each team had a balanced skill set. Developers were only able to complete the first phase in the time allotted. This design allows one to examine programmer's ability to apply formal specification-driven development quickly, and compares early quality differences in the FSDD and SDLC approach.

### **3.2.1.3. SOFTWARE ARTIFACTS**

Teams submitted all of the code that they completed for the project. The code was collected at the end of the semester and evaluated to determine the quality.

### **3.2.1.4. EXPERIMENT**

The independent variable was the use of Formal Specification-Driven Development versus the traditional method of development (SDLC). The dependent variable was software quality. The project had relatively stable and established requirements. In the study, developers were coding with a familiar computer language they were comfortable with. They had the option to choose their development platform. The project, however, included developers with a range of programming experience. While student programmers had similar course backgrounds, they reported a mix of programming backgrounds. Similarly, the project teams ranged from a mix of junior through to more senior developers. As will be discussed, the control and observed groups were balanced in programming experience. Though all the FSDD team members were graduate students, and the control group undergraduates, their programming experiences and college course work were similar.

In the study, confounding factors of requirements unpredictability and technology experience were avoided within each team. Other such factors were circumvented by ensuring consistent language use, stable domain and project assignment, and consistent time frames in the study.

The project is to develop an application that will allow the sale of tickets to different

types of events using the internet. It is designed to be accessed by multiple users simultaneously. The system must accurately display the available tickets for an event. Furthermore, the system shall allow an administrator to create and update the inventory of tickets that are available for sale.

The application will accommodate the following features:

- A user can browse different events that are available
- A user can choose multiple events to purchase
- The events' dates, prices, and venues will be listed
- Events that have been sold-out or canceled will be marked
- A user can add or remove events at any time while browsing
- A message will pop up before a user finalizes a sale
- A notification will be sent via email once a deal is finalized

The initial release of application will not accommodate the following features:

- The system will not display a venue's seating chart
- The system will not allow a user to choose a specific seat for an event
- The system will not be able to perform credit card verification

### 3.2.1.5. FORMAL SPECIFICATION DRIVEN DEVELOPMENT (FSDD) METHOD

To further maintain privacy, student results are reported collectively only. The approval for the study was obtained from Southern Polytechnic State University Institutional Review Board (IRB) prior to conducting the studies. The purpose of the IRB is to regulate all research activities involving human subjects on the campus of Southern Polytechnic State University. The Board ensures that people who participate in research are not treated unethically and are in agreement with all federal and state laws and regulations. Prior to submitting the application, a collaborative institutional training initiative (**CITI**) was completed in Human Subjects Researchers Curriculum and Responsible Conduct of Research for Engineers Curriculum.

The two teams used Microsoft Visual Studio 2013 (Microsoft, 2015) for development. MVC was the preferred architectural style for both the FSDD and the SDLC teams. Visual Studio 2013 takes advantage of its NuGet Package Manager (Microsoft, 2015) to add external tools. TDD/BDD framework that works in C# is SpecFlow (SpecFlow, 2013). SpecFlow was used as the framework of choice, though others are likewise as good. SpecFlow is an open source tool that could be downloaded and installed (SpecFlow, 2013).

The other tool that works seamlessly with Visual Studio is NUnit (NUnit, 2015). It is a free tool that could also be downloaded and installed. NUnit integrates very well with SpecFlow in the .NET environment. NUnit is used for unit testing and adds to the TDD realization. Both SpecFlow and NUnit are added to projects using the Library Package Reference (Microsoft, 2015). The FSDD team installed these tools at the beginning of the

development process. Though SpecFlow can be used for testing, most of the testing was done by NUnit, since it is more compatible with Junit, and the team was more familiar with Junit. The formal specification was done using Z notation. This was a formal specification language of choice used for modeling in the development process.

### 3.2.1.6. SOFTWARE METRICS AND ANALYSIS

The project was a semester-long team project. As a result, the metrics generated and analysis conducted will closely follow that of the previous sections. The project was completed using the ASP and C# integrated development environment, and simple assert statements for automated unit testing. The .NET (Microsoft, 2015) was used to generate some project-level metrics.

**Table 1: C# Code Metrics**

<b>Metric</b>	<b>Expanded Name</b>
DOI	Depth of Inheritance
CC	Class Coupling
MI	Maintainability Index
CCP	Cyclomatic Complexity
LOC	Lines of Code

The “Metric” column in Table 1 gives the equivalent metric abbreviation from the .Net-based experiment. The metrics are deliberated in Appedix 10.

Test volume metrics will be evaluated in the study, but test coverage will not be presented for both teams. The analysis techniques will be the same.

### 3.2.1.7. STATIC CODE ANALYSIS

**Table 2: Sample Metrics by attribute**

<b>Attribute</b>	<b>Metrics</b>
Complexity	Cyclomatic Complexity (CCP) Depth of Inheritance (DOI)
Coupling	Coupling between Objects (CC)
Cohesion	Lack of Cohesion of Methods LOC/Method
Size	LOC/Method LOC/Class LOC/Project
Maintenance	Maintenance Index (MI)

An extensive search produced many static code analysis metrics tools, but the Visual Studio 2013 Code analysis tool was acquired and evaluated for the purposes of this research. Cohesion metric is not utilized in the research and shown above in Table 2 because it is not measured by the Visual Studio Code analysis tool since it is one of the internal quality features. The only attributes listed above can be found in any typical engineering texts since they are the traditional ones. For example, Maintainability Index is a software metric which evaluates the how easy it can be to support and change the source code. It is subsequently calculated through by a fomulae that consists Cyclomatic Complexity, SLOC (Source Lines Of Code) and Halstead volume. It is utilized by several other software tools such as Microsoft Visual Studio 2013 development environment (Lacchia, 2015).

- the derivative utilized by Visual Studio as quoted(Lacchia, 2015).

$$MI = \max[0, 100(171 - 5.2 \ln V - 0.23G - 16.2 \ln L / 171)].$$

Where:

- V is the Halstead Volume (see below);
- G is the total Cyclomatic Complexity;
- L is the number of Source Lines of Code (SLOC);
- C is the percent of comment lines (important: converted to radians).

Maintainability Index should be taken seriously and held in high regard since it is an experimental Index like other metrics. (Lacchia, 2015).

### **Halstead Metrics**

The goal Halstead had was to note the calculatable properties of the software, and their interconnections. These numbers are statically computed from the source code:

- $\eta_1$  = the number of distinct operators
- $\eta_2$  = the number of distinct operands
- $N_1$  = the total number of operators
- $N_2$  = the total number of operands

From these numbers several measures can be calculated:

- Program vocabulary:  $\eta = \eta_1 + \eta_2$
- Program length:  $N = N_1 + N_2$
- Calculated program length:  $\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$
- Volume:  $V = N \log_2 \eta$
- Difficulty:  $D = \eta_1^2 \cdot N_2 \eta_2$
- Effort:  $E = D \cdot V$
- Time required to program:  $T = E / 18$  seconds



- Number of delivered bugs:  $B=V3000$ .

The search was focused on tools that generate metrics from C# code. The static analysis tool comes from Microsoft. The fully functional trial version of Visual Studio 2013 (Microsoft, 2015) was acquired for this analysis. The tool produced many traditional and object-oriented metrics. Table 2 shows the metrics and their categories denoted by “Attribute”. There are other metrics for each of these categories but only the metrics used in the study are included in Table 2. The metrics tools parsed Excel output files that were then consolidated. Project metrics, class and method metrics were obtained using the Visual studio analysis tool (Microsoft, 2015).

### **3.2.1.8. DYNAMIC TEST COVERAGE ANALYSIS**

All software produced was expected to have associated automated unit tests. Code from the *assert()* statements were embedded in the source code, but separated in a global *run\_tests()* function. Code from the experiment utilized the NUnit framework, so the test code was separate from the source/production code. A couple of factors weighed in on the decision not to collect SDLC test coverage metrics. One was that there were no written automated tests for the SDLC code before submission. As a result, it was not reasonable to examine manually the SDLC project to determine what tests were working. Although this is a very doubtful metric, it gives an indication of testing effort.

### **3.2.1.9. ASSESSMENT AND VALIDITY**

Data collected from the experiment were analyzed statistically. The next two chapters will report results of this analysis. A statistical test such as t-Test for Two-Sample Assuming Unequal Variances was employed to determine if differences between the SDLC solution and FSDD solution metrics were statistically significant. The results are only reported in aggregate, as a team effort. In fact, the experiment design and corresponding results should establish the internal validity of the experiment. As mentioned earlier, care was taken to ensure that the control and experimental groups are balanced in terms of their programming skills. Each team used the programming language they were

comfortable with. Both groups were given the same specifications to do the project, to ensure that no bias was introduced. None of the graduate students had an undergraduate degree in either Computer Science or Software Engineering. Also the graduate students, except one of them works in the software development field, in contrast to the a couple of the undergraduates students who had programing jobs. The experiment was slightly skewed towards the SDLC team.

## **4. RESULTS AND ANALYSIS**

This chapter summarizes research conducted with student programmers in a capstone course at the Southern Polytechnic State University. The two projects using FSDD and SDLC will be analyzed in this chapter. The chapter begins with a description of the metrics collected and the corresponding analysis performed. Each project and corresponding results are then described in turn.

### **4.1. METRICS COLLECTION AND ANALYSIS**

The experiment was a semester-long team project. The metrics generated and analysis conducted will closely follow that of Chapter 3. The projects were completed using ASP and C# in an integrated development environment and simple assert statements for automated unit testing, as described in Chapter 3. Students worked in groups of four. Different metrics tools were evaluated for the study, and the.NET analysis tool (Microsoft, 2015) was used to generate the code metrics for the study. Visual Studio, 2013 Analysis tool was also used to produce method, class, and additional project level metrics. Table 1 indicates the metrics that will be used for internal quality measurements. Test volume metrics will be evaluated in the study, but test coverage will not be presented for both teams. The analysis techniques will be the same.

### **4.2. INTERNAL QUALITY RESULTS**

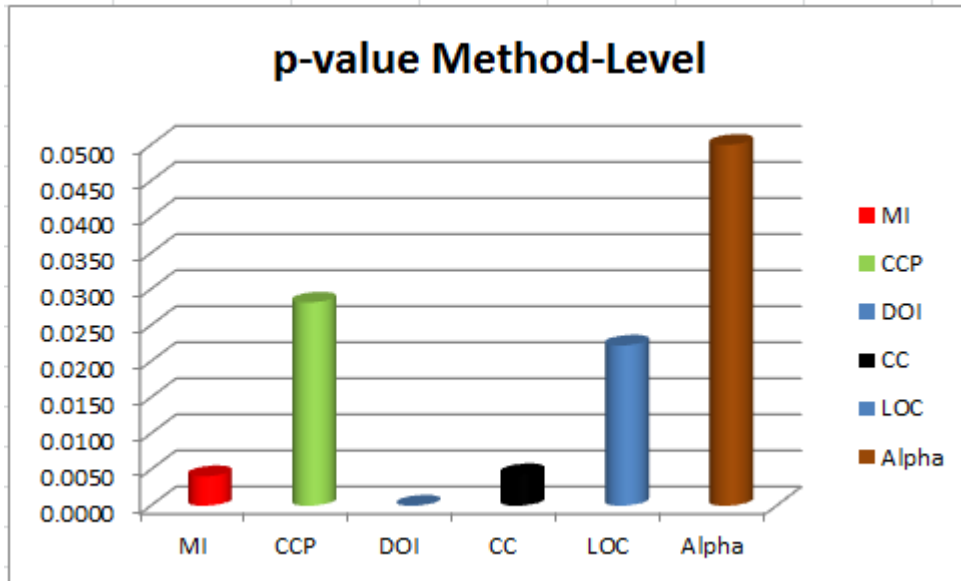
This section reports, describes and compares the internal design quality metric results. The metrics are broken down into the method, class, and project levels.

#### 4.2.1. METHOD-LEVEL METRICS

**Table 3: Summary of Methods Metrics**

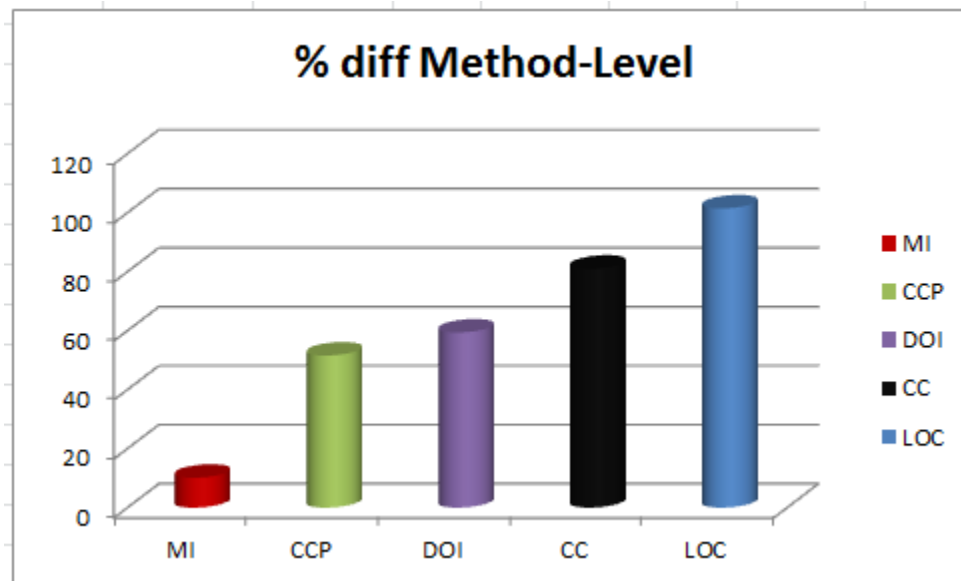
<b>Metric</b>	<b>p-value</b>	<b>Sig?</b>	<b>Higher Method</b>	<b>FSDD Mean</b>	<b>FSDD SDev</b>	<b>SDLC Mean</b>	<b>SDLC Sdev</b>	<b>% diff</b>
MI	0.004094	Yes	FSDD	88.67	12.00	80.12	16.13	10.13
CCP	0.028153	Yes	SDLC	9.45	10.16	16.02	17.92	51.56
DOI	0.000019	Yes	SDLC	1.30	0.68	2.40	1.51	59.39
CC	0.004373	Yes	SDLC	5.02	7.30	11.87	14.71	81.02
LOC	0.022203	Yes	SDLC	15.12	26.58	46.25	90.97	101.45

This section presents the results of the method-level analysis of the two teams, using FSDD and SDLC with ASP and C# in the spring 2015 study. Table 2. gives a summary of all the metrics for the methods in the two techniques. The p-values as compared to the alpha value (0.05) are shown in Figure 5.



**Figure 5: Method-Level Metrics p-values**

All the metrics indicate substantial statistical significance. The percent differences are also crucial. The percentage differences range from 101% for the line of code (LOC) to 10% for the Maintainability Index (MI).



**Figure 6: % difference in Method-Level Metrics**

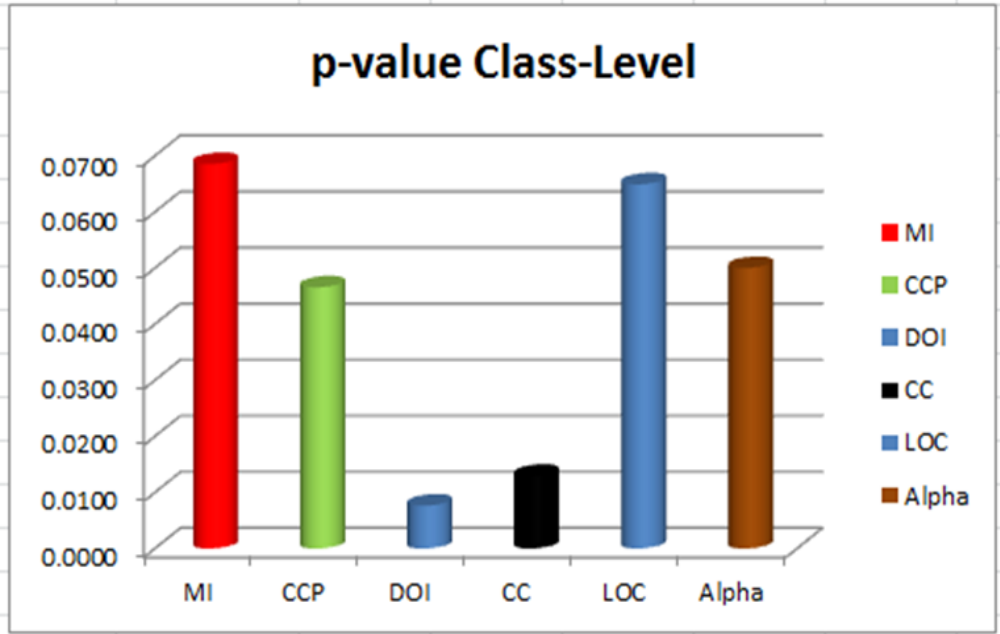
Those differences are depicted in Figure 6. FSDD code has significantly desirable Cyclomatic Complexity (CC), Depth of Inheritance (DOI), Class Coupling (CC) and Line of Code (LOC). LOC with a value of 101% is so significant compared to the other values. None of the p-values comes close to the alpha value of .05. The data results for the method-level metric infers that the FSDD technique may be more likely to produce smaller solutions (LOC). The solutions are less complex (CCP), less cohesive (CC), and easier to maintain (MI). FSDD code has a significantly higher internal quality than SDLC code considering method-level metrics.

#### 4.2.2. CLASS-Level Metrics

**Table 4: Summary of Class-level Metrics**

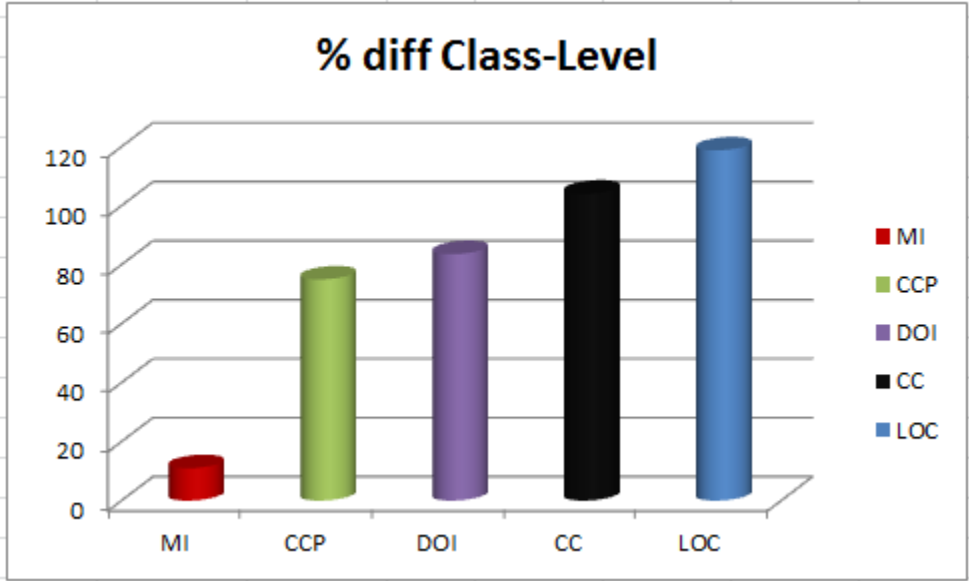
Metric	p-value	Sig?	Higher Method	FSDC Mean	FSDC SDev	SDLC Mean	SDLC Sdev	%diff
MI	0.068531	No	FSDC	89.09	11.37	79.80	13.71	11.00
CCP	0.046549	Yes	SDLC	36.00	41.60	79.20	81.76	75.00
DOI	0.007635	Yes	SDLC	1.27	0.65	3.10	1.66	83.58
CC	0.012956	Yes	SDLC	11.64	15.21	36.90	24.35	104.10
LOC	0.064904	No	SDLC	57.64	74.43	226.20	247.09	118.78

Table 4 gives a summary of all the class-level metrics for the two techniques. This section reports the class-level metrics analyzed for the study. The results show four of the metrics with significant statistical p-value. Two of the metrics are not statistically viable when comparing their p-values to the alpha value (0.05).



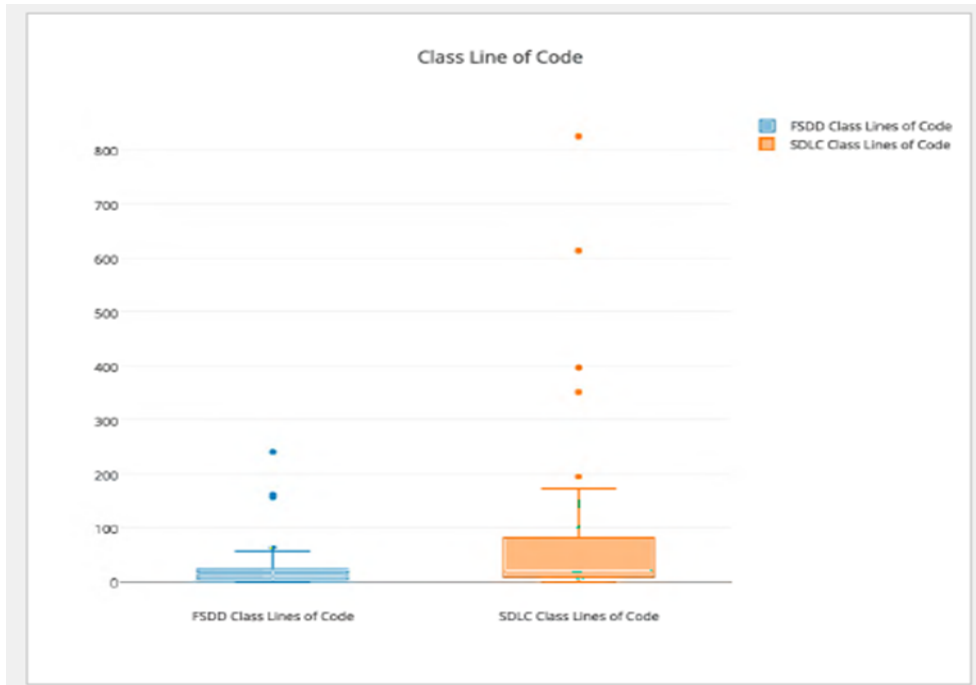
**Figure 7: Class-Level Metrics p-values**

Figure 10 shows the side-by-side comparison of all the p-values and alpha. LOC and MI are the statistically insignificant metrics. The percent differences are all significant except MI. LOC is interesting in that it was not statistically significant, as seen in Figure 11. LOC yields the greatest percentage difference (119%).





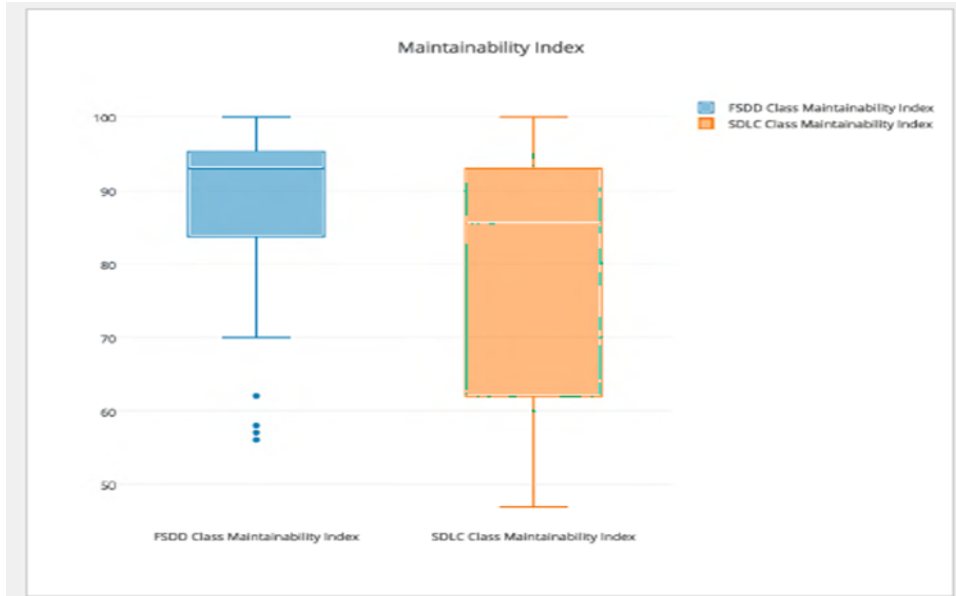
**Figure 8: % Difference in Class-Level Metrics**



**Figure 9: Box plot for line of code in Classes**

Figure 12 shows the box plot of the class-level LOC. There are some outliers on this plot that may be the reason we have very high percentage difference in LOC but statistical p-value that made it insignificant. The outliers are many in the LOC of the FSDD box plot. The same reason can be used for the box plot of the maintainability index MI of Figure 12. Though the percentage difference is not much, there are also outliers in this case, thereby skewing the results.

Again, just like the method-level, the class-level data portrays statistically significant data in favor of FSDD code as having a higher internal quality compared to SDLC code.



**Figure 10: Box plot Class-Level Maintainability Index**

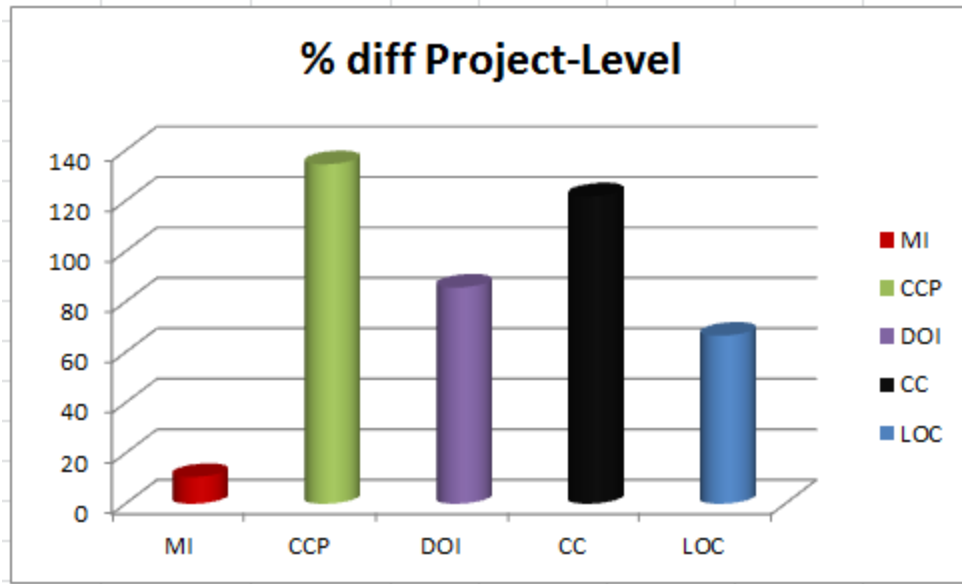
The results signify some trends that can be identified. For instance, the SDLC class tends to be larger and more complex (LOC). The SDLC software tends to use more of a procedural approach. These concerns are reflected in the CC and LOC measures. The results demonstrate that the FSDO technique may be more likely to produce smaller solutions (LOC) that are less complex (CCP), less cohesive (CC), and easier to maintain (MI).

### 4.2.3. PROJECT-LEVEL METRICS

Table 5: Project-Level Metrics results

Metric	Sig?	Higher Method	FSDD	SDLC	% diff
MI	No	FSDD	89.00	80.00	10.65
CCP	Yes	SDLC	163.00	833.00	134.54
DOI	Yes	SDLC	2.00	5.00	85.71
CC	Yes	SDLC	51.00	210.00	121.84
LOC	Yes	SDLC	1203.00	2405.00	66.63

The project-level metrics are reported in Table 5. From this table, one notices that the SDLC projects tend to be larger (LOC) and more complex (CCP). The FSDD project is less complex (CCP), but the discrepancy is not much. The Cyclomatic Complexity (CCP) shows high significant percentage difference. The code Maintenance Index (MI) was not that significant. Both the FSDD and SDLC solutions are between the 20 and 100 interval of favorable values. The Project-level metrics shows the most significant percentage differences compared to the class-level and method-level metrics. The entire project being considered as an entity makes the FSDD solution even more desirable.



**Figure 11: % Difference Project-level Metrics**

The CCP difference (134.54%) is so significant, since it trumps that of the class-level and method-level. The results point to the fact that when the project is considered as one big giant entity, the complexity of the SDLC solution becomes even greater. This complexity increase can also be seen in the CC percent difference value (121.84%). The CCP and the CC are metrics of complexity and will also indicate a lot of tests achieve good code coverage (Microsoft, 2015).

### 4.2.3.1. STATIC CODE ANALYSIS

**Table 6: FSDD Project-level Static Code Analysis results**

Bugs	Name	# Matches	Elements	Group
Warning Critical	Potentially dead Methods	1	methods	Dead Code
warning	Avoid namespaces with few types	1	namespaces	Design
warning	Static fields should be prefixed with a 's_ '	1	fields	Naming Conventions
warning	Avoid methods with name too long	2	methods	Naming Conventions
warning	Class with no descendant should be sealed if possible	2	types	Object Oriented Design
warning	A stateless class or structure might be turned into a static type	2	types	Object Oriented Design
warning	Non-static classes should be instantiated or turned to static	2	types	Object Oriented Design
warning	Methods should be declared static if possible	6	methods	Object Oriented Design
warning	Don't assign static fields from instance methods	1	fields	Object Oriented Design
warning	Mark assemblies with CLS Compliant	1	assemblies	System
warning	Methods that could have a lower visibility	6	methods	Visibility
warning	Types that could have a lower visibility	1	types	Visibility

Table 6 shows the static code analysis of the code developed using FSDD. It indicates twelve warnings, of which only one is of a critical nature. The table portrays no code quality issues with the FSDD solution. Looking at Table 7 with the results of the SDLC static code analysis, there are 46 warnings, of which six are critical and nine have quality issues.

Tables 6 and 7 show a marked difference in the static code quality between FSDD and SDLC. It signifies the superior quality of the FSDD approach and further tips the scale towards this technique.

**Table 7: SDLC Project-level Static Code Analysis results**

Bugs	Name	# Matches	Elements	Group
warning	Assemblies with poor cohesion (Relational Cohesion)	1	assemblies	Architecture and Layering
warning	Avoid namespaces dependency cycles	1	namespaces	Architecture and Layering
Warning Critical	Avoid namespaces mutually dependent	1	namespaces	Architecture and Layering
warning	UI layer shouldn't use directly DAL layer	27	types	Architecture and Layering
warning	UI layer shouldn't use directly DB types	9	types	Architecture and Layering
warning	Methods potentially poorly commented	17	methods	Code Quality
warning	Methods too big	13	methods	Code Quality
warning	Methods with too many local variables	4	methods	Code Quality
warning	Methods with too many parameters	8	methods	Code Quality
warning Critical	Methods with too many parameters - critical	4	methods	Code Quality
warning	Quick summary of methods to refactor	25	methods	Code Quality
warning	Types with poor cohesion	3	types	Code Quality
warning	Types with too many fields	6	types	Code Quality
warning	Types with too many methods	1	types	Code Quality
Warning Critical	Potentially dead Fields	144	fields	Dead Code
Warning Critical	Potentially dead Methods	6	methods	Dead Code
warning	Avoid namespaces with few types	7	namespaces	Design
warning	Declare types in namespaces	1	namespaces	Design
warning	Instances size shouldn't be too big	22	types	Design
warning	Nested types should not be visible	2	types	Design
Warning Critical	Avoid having different types with same name	2	types	Naming Conventions
warning	Avoid naming types and namespaces with the same identifier	1	types	Naming Conventions
warning	Instance fields should be prefixed with a 'm_'	296	fields	Naming Conventions
warning	Methods name should begin with an Upper character	14	methods	Naming Conventions
warning	Static fields should be prefixed with a 's_'	12	fields	Naming Conventions
warning	Class with no descendant should be sealed if possible	48	types	Object Oriented Design
warning	Methods should be declared static if possible	25	methods	Object Oriented Design
warning	Non-static classes should be instantiated or turned to static	34	types	Object Oriented Design
Warning Critical	Don't assign a field from many methods	2	fields	Purity - Immutability - Side-Effects
warning	Fields should be marked as Read Only when possible	9	fields	Purity - Immutability - Side-Effects
warning	Structures should be immutable	2	types	Purity - Immutability - Side-Effects
warning	Avoid defining multiple types in a source file	1	types	Source Files Organization
warning	Namespace name should correspond to file location	71	types	Source Files Organization
warning	Types declared in the same namespace, should have their source files stored in the same directory	2	namespaces	Source Files Organization
warning	Types with source files stored in the same directory, should be declared in the same namespace	2	namespaces	Source Files Organization
warning	Mark assemblies with assembly version	1	assemblies	System
warning	Mark assemblies with CLS Compliant	1	assemblies	System
warning	Mark assemblies with Com Visible	1	assemblies	System
warning	Avoid public methods not publicly visible	37	methods	Visibility
warning	Avoid publicly visible constant fields	2	fields	Visibility
warning	Event handler methods should be declared private	76	methods	Visibility
warning	Fields should be declared as private	275	fields	Visibility
warning	Fields that could have a lower visibility	275	fields	Visibility
warning	Methods that could have a lower visibility	403	methods	Visibility
warning	Types that could have a lower visibility	50	types	Visibility

### **4.3. TEST RESULTS**

This section presents the test density and coverage measurements for the study. It should be recalled that the SDLC team wrote no automated tests, so they are not included in this discussion, consequently there are no data for comparison. The results from the FSDD team will be combined in the final section of this chapter. FSDD teams achieved a very high test coverage metrics at 90% for black box and 98% for white box testin

## **5. EVALUATION, OBSERVATION, AND DISCUSSION**

This final chapter will summarize and evaluate the results of this research.

Observations will be drawn, and possible conclusions will be proposed. Future work will also be identified. This research makes several substantial contributions. Foremost is the empirical evidence regarding the effects of Formal Specification-Driven Development on internal software quality. Section 5.1 will summarize this evidence and categorize it in terms of the desirable quality attributes identified in the earlier chapter. This evidence provides compelling motivation to adopt FSDD to reduce code size and complexity, and increase programmer testing and testability. The evidence also raises some interesting questions about how FSDD affects coupling, cohesion, and maintainability.

The research is the first significant examination of the effects of FSDD on internal software quality. As such, it creates a benchmark to be reviewed and assessed. This work provides a basis for conducting replicated studies in similar environments that will reinforce and clarify these results. Finally, the last section will summarize this work and recommend future directions for related research.

### **5.1. EMPIRICAL EVIDENCE OF FORMAL SPECIFICATION-DRIVEN DEVELOPMENT**

#### **EFFICACY**

The main contribution of this research is the empirical proof of the effects on internal software quality, applying FSDD technique in the software development process.

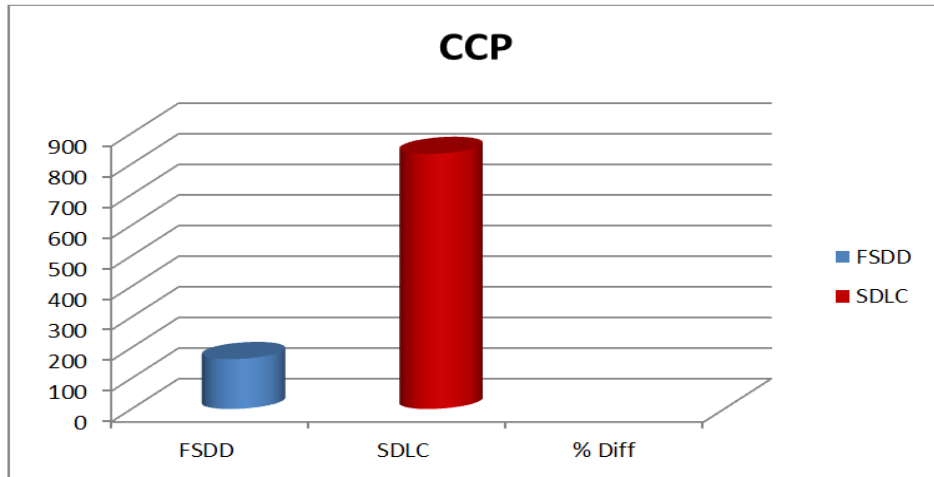


Chapter 4 presented a high volume of empirical data, along with some analysis. This section will summarize this data and reexamine the initial hypotheses. Data will be grouped and visualized with bar charts to accommodate drawing conclusions. The longer the bar, the larger the difference between the FSDD and SDLC projects on that particular metric. Special attention will have to be paid to whether larger values are desirable or not. For instance, with a maintainability index, larger values are more desirable. However with complexity metrics, smaller values are more desirable.

The first section will focus on the substantial improvements that the FSDD approach has on software testing. The following sections will consider complexity, coupling, cohesion, and size metrics, and then combine them to examine the effects of FSDD on the four desirable software characteristics of understandability, maintainability, and testability.

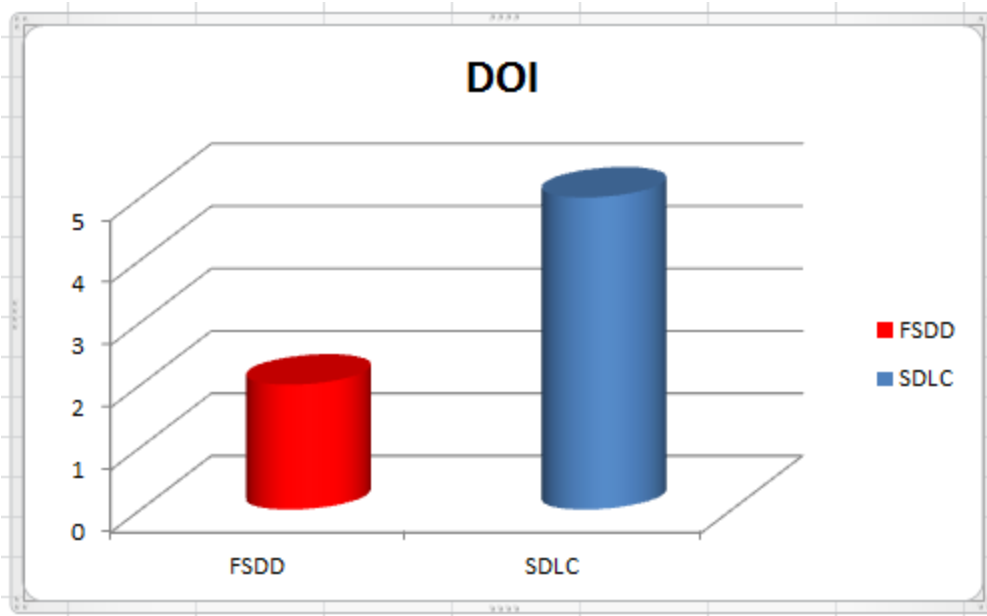
### **5.1.1. QUANTITATIVE EVIDENCE: COMPLEXITY**

Figure 16 displays the differences in cyclomatic complexity metrics between the FSDD and SDLC projects for the study. In all of the complexity metrics, lower values are more desired. The complexity figures tell an interesting story. It appears that developers tend to write less complex software when using the FSDD approach. However, more developers tend to write more complex code with the SDLC approach.



**Figure 12: Difference in Project-Level Cyclomatic Complexity**

Perhaps the influence of experience with the FSDD approach provides an enduring effect that extends through future projects. There is also another metric to be looked at when considering code complexity, and this is the depth of inheritance DOI.

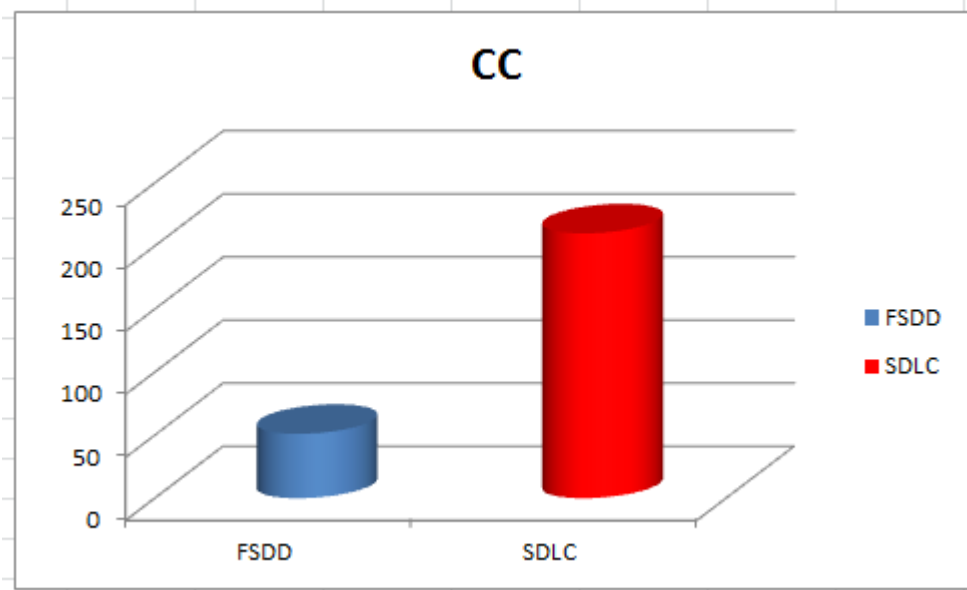


**Figure 13: Difference in Depth of Inheritance**

Figure 17 shows the significant difference between SDLC and FSDD in the way the number of class creations extend to the root of the class pyramid. The longer the extension, the more complex the code, and also the more difficulty understanding the code.

Many of these differences were statistically significant in both the method and class-levels. Figure 16 and Figure 5 report that differences were statistically significant at  $p < .05$ . A 'Yes' in a cell indicates that the metric was significant for that experiment.

### 5.1.2. QUANTITATIVE EVIDENCE: COUPLING

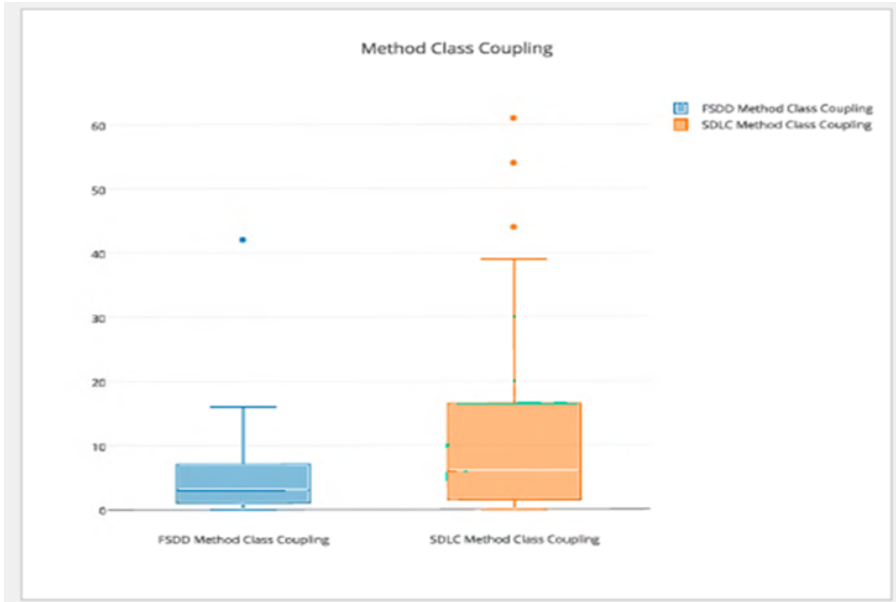


**Figure 14: Difference in Project-Level Cyclomatic Complexity**

Figure 18 displays the differences in class coupling metrics between the FSDD and SDLC projects for all of the experiments in which the typical solution contained at least two objects. For both coupling metrics, lower values are more appropriate.

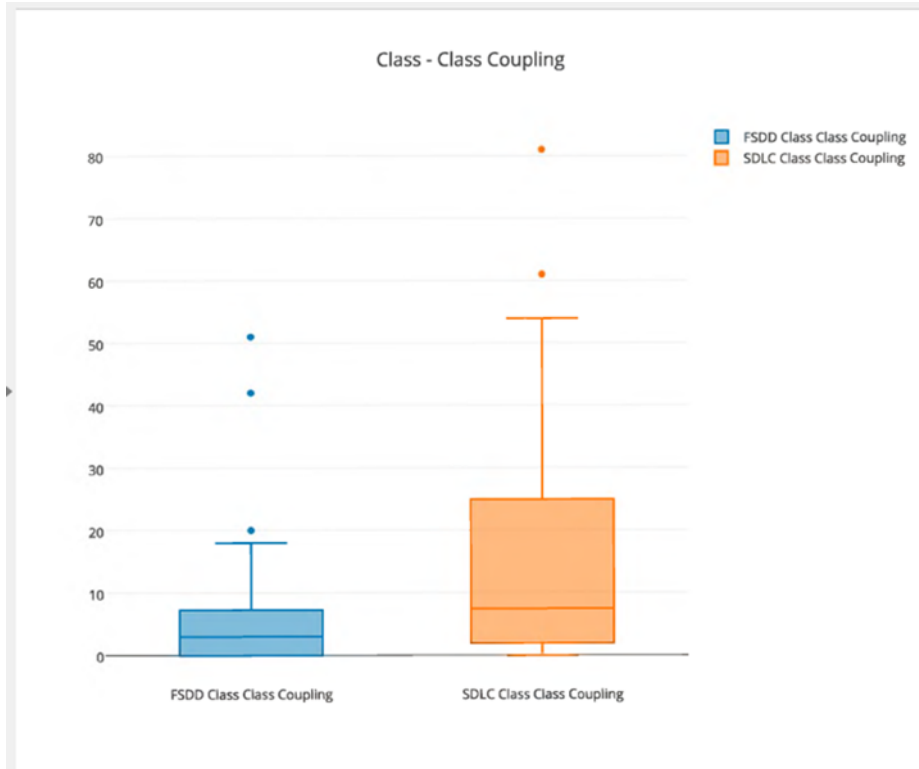
This chart indicates that the FSDD approach decreases coupling. The results show a statistical significance. Thus, we can claim that the FSDD approach reduces class coupling, since it is statistically significant. The FSDD approach seems to cause developers to write smaller, less complex methods and classes. More connections between these units may result.

An interesting question is whether the increased coupling is good or bad. Coupling can be inappropriate when it is inflexible, and changes in one module cause changes in another module. However, it can be argued that some coupling can be useful, particularly when the coupling is either constituted or uses abstract connections such as interfaces or abstract classes. Such code can be considered highly flexible and thus more maintainable and reusable.



**Figure 15: Box plot of Method-Level Class Coupling**

The box plot of the coupling in method-level of Figure 19 and class-level of Figure 20 clearly backs this assertion that FSDD yields less coupled code. It is not difficult to draw this conclusion regarding coupling in this experiment. There are overwhelming indications that the FSDD approach decreases coupling, although an increase could have indicated a desirable type of coupling through abstractions. The differences are shown in Figure 19 and Figure 20 box plots for the class-level and method-level class coupling.



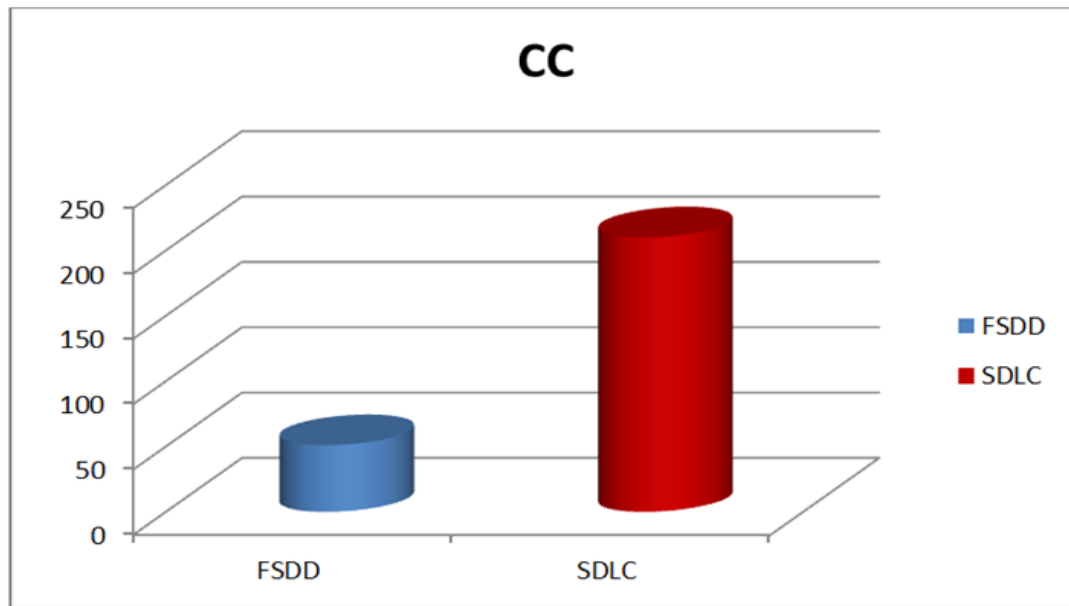
**Figure 16: Box plot of Class-Level Class Coupling**

Statistically significant: so much can be said with confidence. In contrast to the complexity metrics, these results do not necessarily reject the RQ Null Hypothesis.

### 5.1.3. QUANTITATIVE EVIDENCE: COHESION

Like the coupling measures, the empirical results are very apparent regarding the effects of the FSDD/SDLC approach on cohesion. Attempts at determining trends led to two charts discussed here. Figure 17 reports the differences in the class coupling (CC) metric for the experiment in which the typical solution will have more than one object. Compared to most metrics reported, lower class coupling (CC) values are desirable, indicating better

cohesion. Reasonable explanations for the exceptions seem harder to come by in the case of cohesion.



**Figure 17: Differences in Cohesion Metric CC**

One might expect the SDLC/library project of the experiment to have more cohesion than the FSDD application. The differences were statistically significant, so perhaps there is nothing that can be said about the effects of the FSDD/SDLC approach on cohesion. In the study, the FSDD project had more methods and classes.

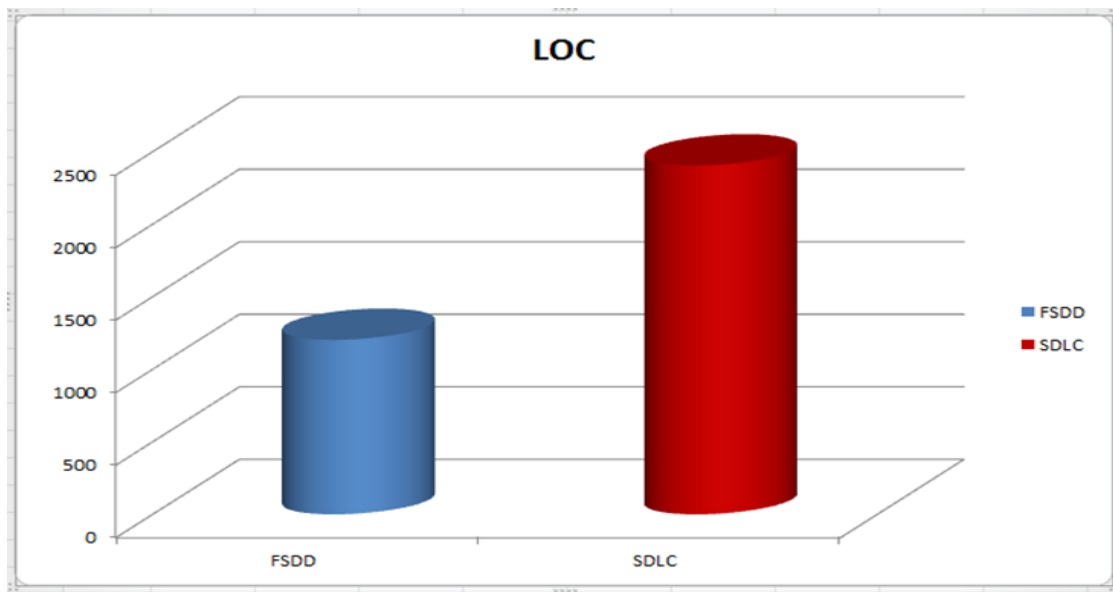
The comparison with the number of classes makes sense because the projects within the research were solutions to the same problem. The greater number of methods and classes with the FSDD approach was anticipated, as smaller units are more testable. But the corresponding decrease in cohesion is perhaps surprising. One might expect

solutions with more classes to have smaller and more cohesive classes. However, this seems not to be the case. The differences in the number of methods and number of classes in the projects were statistically significant.

Like the coupling measures, there are some indications that the FSDD approach may decrease cohesion. However, differences were statistically significant. As a result, the cohesion metrics do lend support to accepting the RQ null hypothesis.

#### 5.1.4. QUANTITATIVE EVIDENCE: SIZE

This section considers differences in software size metrics. Figure 18 compares LOC for the two approaches in the experiment, in which the typical solution contained the objects.



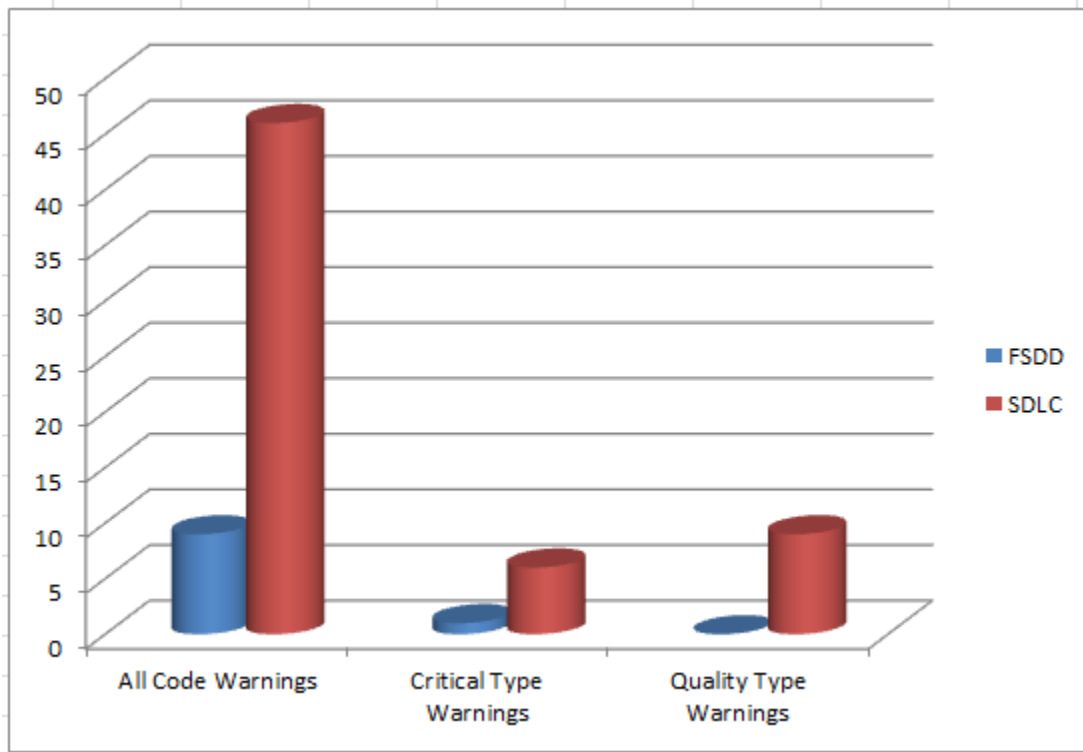
**Figure 18: % Difference in Size Metrics**



The chart reveals the trend that SDLC developers tend to write larger methods and classes. We see here that FSDD developers consistently implemented more classes and methods with more variables, and that the total number of statements in a solution reversed in favor of smaller FSDD solutions. The number of methods and classes was statistically significant. The code size metrics and lines of code used in the study have often been criticized (Murphy & Stone, 1995), but they are beneficial in some situations. Less code is more maintainable compared to a complex one. Smaller modules are more reusable and testable. These results indicate that the FSDD approach seems to influence developers to write smaller methods and classes.

## 5.1.5. QUANTITATIVE EVIDENCE: STATIC CODE

### QUALITY ANALYSIS



**Figure 19: Static Code Quality**

Figure 18 gives a summary of the static code quality analysis of the study. There is a big discrepancy between the number of warnings produced by the solutions of SDLC and FSDD. The differences are also seen in both the critical and quality type warnings. In fact, the FSDD code had no quality issues, while the SDLC code had nine. Observations from earlier evidence strengthen the notion that FSDD produces better internal quality code than SDLC.

## **5.1.6. EMPIRICAL EVIDENCE SUMMARY AND CONCLUSIONS**

Complexity, coupling, cohesion and size were identified as relevant components of the quality characteristics: understandability, maintainability, reusability, and testability.

**Table 8: Project-Level Metrics results**

<b>Metric</b>	<b>Higher Method</b>	<b>FSDD</b>	<b>SDLC</b>	<b>% diff</b>	<b>Desirable</b>
MI	FSDD	89	80	10.65	FSDD
CCP	SDLC	163	833	134.54	FSDD
DOI	SDLC	2	5	85.71	FSDD
CC	SDLC	51	210	121.84	FSDD
LOC	SDLC	1203	2405	66.63	FSDD

Table 8 summarizes the results in these categories from the previous sections. The table reports that SDLC method had desirable values. The FSDD method produced more appropriate values for the analogous experiment and characteristic. The FSDD approach provides more desirable values as opposed to SDLC. Blank cells indicate that results were not valid or available for the research. Almost all the metric differences were statistically significant.

It appears that the FSDD approach did improve internal software quality for the FSDD team in terms of complexity, size, and testing. The evidence is significant enough to make the following claim. Developers applying the FSDD approach are likely to write smaller units (methods and classes) than they would write with an SDLC approach.

There was a more favored approach in terms of coupling and cohesion. It appears that an FSDD approach may be best in terms of complexity, coupling, and cohesion. Coupling, cohesion, complexity, and size were identified as components of the desirable quality characteristics of understandability, maintainability and reusability. The claim cannot be made that the FSDD approach improves all of the features entirely. Hence, we cannot reject the RQ Null Hypothesis. However, this research has demonstrated that the FSDD approach can cause significant internal code quality improvements by lowering code complexity and reducing the size of methods and classes. Combined with the improvements in complexity and size, this provides a compelling incentive for developers to consider adopting FSDD.

## **5.2. SUMMARY AND FUTURE WORK**

Despite many significant advances, software construction is still plagued with many failures. Development organizations struggle to adopt smart development methods, due to a lack of empirical evidence of what methods are best in which circumstances. While some individual programmers and organizations have learned to value and apply well-organized, yet flexible methods, students do not graduate with these skills.

Formal Specification-Driven Development is a disciplined development practice that promises to improve software design quality while reducing defects, with no increased effort. This research carefully examined the possibility of FSDD to deliver these benefits. This research has demonstrated that FSDD can and is likely to improve some software quality aspects at minimal cost over a comparable SDLC approach. In particular, it has shown significant differences statistically in the areas of code complexity, size, and maintainability. These internal quality differences can substantially decrease software defects. Additional empirical studies should replicate the study in similar and new settings.

Future studies should examine if the use of C# and JUnit improves FSDD acceptance and efficacy in programming courses. Future studies could examine the question of how much up-front software architecture and design work should ideally be finished before engaging in the FSDD process. These studies should consider scale and safety concerns of the projects.

Another suggestion would be to consider the learning curve of the FSDD approach as well as programmer discipline with the FSDD approach in practice. Some of the students in the study noted the high level of discipline required to stay with the FSDD approach on a daily basis. The FSDD team indicated they would be keen to use this new method, but not within limited time constraints, as was the case in the semester-long project.

This study compared one FSDD team to one SDLC team, due to the limited number of students in the spring 2015 capstone class. Future studies should examine their efficacy as it applies to a broad cross-section, probably three to four groups per approach.

As a result, it is believed that this research can have a significant impact on the software development process. FSDD may in an indirect way transform the methods by which we develop software. Some software development organizations will be convinced to adopt FSDD in appropriate situations. New textbooks can be written applying the FSDD learning approach. As students learn to use this new and more methodical approach to software development, they will carry this into the future, and this will impact the way software is developed.

## 6. BIBLIOGRAPHY

- Aichernig, B. K., Lorber, F. & Tiran, S. (2012). Formal test-driven development with verified test cases. Retrieved 8/20/2014 from [https://online.tugraz.at/tug\\_online/voe\\_main2.getVollText?pDocumentNr=275810&pCurrPk=67400](https://online.tugraz.at/tug_online/voe_main2.getVollText?pDocumentNr=275810&pCurrPk=67400)
- Alawneh, S. G. & Peters D. K. (2013). Using test oracles and formal specifications with test-driven development. *International Journal of Software Engineering & Knowledge Engineering* 23(3): 361-385. Retrieved 8/20/2014 from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.416.9592>
- Ambler, S. W. (2007). Test-driven development of relational databases. *IEEE Software* 24(3): 37-43. Retrieved 09/3/2014 from <https://connect.spsu.edu/eds/command/,DanaInfo=.aeuCfEki0lys0570s54+detail?sid=060fd54b-67a6-44a4-b215-d357ef58d497%40sessionmgr115&vid=6&hid=113>
- Baumeister, H. (2004). Combining formal specifications with test driven development. *Lecture Notes in Computer Science*, 3134, 1-12. Retrieved 8/30/2014 from <http://www.pst.ifi.lmu.de/~baumeist/publications/baumeister04a.pdf>
- Boehm, B. & Basili, V. R. (2001). Software Defect Reduction Top 10 List. Retrieved 6/29/2015 from <http://cs.umd.edu/~basili/publications/journals/J81.pdf>
- Beck, K. (2001). Aim, fire (test-first coding). *Software, IEEE*, vol.18, no.5, pp.87,89, Sep/Oct 2001. Retrieved 8/30/2014 from <https://connect.spsu.edu/stamp/,DanaInfo=.aifgh1urvznJtqrsO48y+stamp.jsp?tp=&ar number=951502>
- Beck, K., Gamma E. (1998). Test Infected: Programmers Love Writing Tests. *Java Report*, vol. 3, pp. 51-56, 1998. Retrieved 10/30/2014 from [http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fwww-public.int-evry.fr%2F~gibson%2Fteaching%2FCSC7302%2FReadingMaterial%2FBeckGamma00.pdf&ei=vcpeVLDpGjXasASp1IDgBQ&usq=AFQjCNEwCd4lfy6UgeLi\\_qgvQEHuDq3Cjg&bvm=bv.79189006,d.cWc](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fwww-public.int-evry.fr%2F~gibson%2Fteaching%2FCSC7302%2FReadingMaterial%2FBeckGamma00.pdf&ei=vcpeVLDpGjXasASp1IDgBQ&usq=AFQjCNEwCd4lfy6UgeLi_qgvQEHuDq3Cjg&bvm=bv.79189006,d.cWc)
- Boehm, B. (1987). Industrial software metrics top 10 list. *Computer*, January 2001, pp. 135-137. Retrieved 10/30/2014 from <http://www.cs.cmu.edu/afs/cs/academic/class/17654-f01/www/refs/BB.pdf>
- Carvalho R., Soares Manhães R., and de Carvalho\_F.L., Filling the Gap between Business Process Modeling and Behavior Driven Development, CoRR, 2008. Retrieved 10/2/2014 from <http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CEQQF>

[jAD&url=http%3A%2F%2Fwww.confenis2011.aau.dk%2FdigitalAssets%2F31%2F31415\\_proceedings---short-papers.pdf&ei=XS5pVID0FYynNpnLgYAJ&usg=AFQjCNEzWs9KiX1XJ0nm1nxFXYm-gUCbkw&sig2=10yVsOAqMjDt-f9Gz\\_Q8qw&bvm=bv.79142246,d.eXY](http://www.confenis2011.aau.dk/digitalAssets/31/31415_proceedings---short-papers.pdf&ei=XS5pVID0FYynNpnLgYAJ&usg=AFQjCNEzWs9KiX1XJ0nm1nxFXYm-gUCbkw&sig2=10yVsOAqMjDt-f9Gz_Q8qw&bvm=bv.79142246,d.eXY)

Code Metrics Results: what is it for? (n.d.). Retrieved 5/24/2015 from <https://social.msdn.microsoft.com/Forums/en-US/95897686-485b-4492-a9f0-8a8a83656>

Edwards S. H. (2003). Using test-driven development in the classroom: providing students with automatic, concrete feedback on performance. In *Proceedings of the International Conference on Education and Information Systems: Technologies and Applications EISTA'03*, August. Retrieved 8/30/2014 from <http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fwww.cs.tufts.edu%2F~nr%2Fcs257%2Farchive%2Fstephen-edwards%2Fautomated-feedback.pdf&ei=D50sVJyGJY6PNsetgLGH&usg=AFQjCNELCYQtBlk-IOdhDvf9o-zGH0MwRQw&bvm=bv.76477589,d.eXY>

Erdogmus, H., Morisio, M. & Torchiano, M. (2005). On the effectiveness of the test-first approach to programming. *Software Engineering, IEEE Transactions on*, vol.31, no.3, pp.226,237, March 2005. Retrieved 8/30/2014 from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1423994&isnumber=30744>

Falco, L. (2013) Behavior Driven Development - YouTube. (n.d.). Retrieved 6/29/2015 from <https://www.youtube.com/watch?v=mT8QDNNhExg>

Fucci D., Turhan B.A. (2013). Replicated Experiment on the Effectiveness of test-first development, *Empirical Software Engineering and Measurement. 2013 ACM / IEEE International Symposium on*, vol., no., pp.103, 112, 10-11 Oct. 2013. Retrieved 8/30/2014 from <https://connect.spsu.edu/stamp/,DanaInfo=.aifgh1urvznJtqrsO48y+stamp.jsp?tp=&arnumber=6681343>

Gamma, E. & Beck, K. (2006). JUnit. Retrieved 9/3/2014 from [https://www.google.com/?gws\\_rd=ssl#q=junit+Gamma+pdf](https://www.google.com/?gws_rd=ssl#q=junit+Gamma+pdf)

George B. & Williams L. (2003). An Initial Investigation of Test Driven Development in Industry, *Proc. ACM Symp. Applied Computing*. Retrieved 10/30/2014 from <http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fcollaboration.csc.ncsu.edu%2Fflaurie%2FPapers%2FTDDperv8.pdf&ei=ks1eVMe50tP7sASevILoDQ&usg=AFQjCNHk6TJnNC32UGD8cN65EWGjoQkTBA>

JUnit.org, [www.junit.org](http://www.junit.org), 2004



*Junit Tutorial PDF Book - Asaha.com.* (n.d.). Retrieved from <http://asaha.com/ebooks/junit-tutorial.html> br

Kaufmann R. & Janzen D. Implications of test-driven development: a pilot study. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, pages 298-299. ACM Press, 2003. Retrieved 8/30/2014 from <http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fwww.cs.tufts.edu%2F~nr%2Fcs257%2Farchive%2Fstephen-edwards%2Fautomated-feedback.pdf&ei=D50sVJyGJY6PNsetgLgH&usg=AFQjCNELCYQtBk-IOdhDvf9o-zGH0MwRQw&bvm=bv.76477589,d.eXY>

Keogh E., BDD: A Lean Toolkit. In *Processings of Lean Software & Systems Conference, ASDLC anta, 2010*. Retrieved 8/30/2014 from [http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fwww.djaa.com%2Fsites%2Fdefault%2Ffiles%2Flean\\_ssc\\_2010\\_proceedings.pdf&ei=DDBpVLjYEsuYNvaYgoAK&usg=AFQjCNEI7t43\\_97Pqq-VceoXXhiH0mjjkg&sig2=DKJV\\_Fo1iIDwj\\_UkpyGb4w&bvm=bv.79142246,d.eXY](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fwww.djaa.com%2Fsites%2Fdefault%2Ffiles%2Flean_ssc_2010_proceedings.pdf&ei=DDBpVLjYEsuYNvaYgoAK&usg=AFQjCNEI7t43_97Pqq-VceoXXhiH0mjjkg&sig2=DKJV_Fo1iIDwj_UkpyGb4w&bvm=bv.79142246,d.eXY)

Janzen D.S. & Saiedian, H. (2008). Does Test-Driven Development really improve software design quality? *Software, IEEE, 25(2)*, 77-84. Retrieved 9/3/2014 from <https://connect.spsu.edu/eds/command/.DanaInfo=.aeeuCfEki0lys0570s54+detail?sid=060fd54b-67a6-44a4-b215-d357ef58d497%40sessionmgr115&vid=26&hid=113>

Kumar, S., & Bansal, S. (2013). Comparative study of test driven development with traditional techniques. *International Journal of Soft Computing & Engineering, 2013, Vol. 3, Issue 1*, p.352. Retrieved 9/2/2014 from <http://www.doaj.org/doaj?func=openurl&genre=article&issn=22312307&date=2013&volume=3&issue=1&spage=352>  
<http://www.ijscce.org/attachments/File/v3i1/A1351033113.pdf>

Lacchia, M. (2015). Introduction to Code Metrics. Retrieved 8/8/2015 from <http://radon.readthedocs.org/en/latest/intro.html#maintainability-index>

Lazăr I., Motogna S., and Pârș B., (2010). Behaviour-Driven Development of Foundational UML Components. *Electronic Notes in Theoretical Computer Science 264*, no. 1 (August): 91-105,. Retrieved 9/12/2014 from [http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CDsQFjAD&url=http%3A%2F%2Fsdq.ipd.kit.edu%2Ffileadmin%2Fuser\\_upload%2Fsdq%2Fconferences%2FFESCA2010%2FSimonaMotogna.pdf&ei=hjFpVKL5EcbOggTThYPoAg&usg=AFQjCNFqs0EZfyQepbu3XXsqC3O1Ti-lBg&sig2=7pS0wirHvHjvLkpsi9wqdw&bvm=bv.79142246,d.eXY](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CDsQFjAD&url=http%3A%2F%2Fsdq.ipd.kit.edu%2Ffileadmin%2Fuser_upload%2Fsdq%2Fconferences%2FFESCA2010%2FSimonaMotogna.pdf&ei=hjFpVKL5EcbOggTThYPoAg&usg=AFQjCNFqs0EZfyQepbu3XXsqC3O1Ti-lBg&sig2=7pS0wirHvHjvLkpsi9wqdw&bvm=bv.79142246,d.eXY)

Mark Doliner. Cobertura, 2006. Retrieved 12/20/2014 from <http://cobertura.sourceforge.net/>.

- Microsoft . Code Metrics Values, 2015. Retrieved 5/22/2015 from <https://msdn.microsoft.com/en-us/library/bb385914.aspx>
- Microsoft. NuGet Package Manager for Visual Studio 2013, 2015. Retrieved 5/26/2015 from <https://visualstudiogallery.msdn.microsoft.com/4ec1526c-4a8c-4a84-b702-b21a8f5293ca>
- Microsoft. Visual Studio 2013, 2015. Retrieved 4/20/2015 from <https://msdn.microsoft.com/query/dev12.query?appId=Dev12IDEF1&l=en-US&k=k%28MSDNSTART%29&rd=true>
- Muhammad Shahid, Suhaimi Ibrahim, and Mohd Naz'ri Mahrin, (2011). A Study on Test Coverage in Software Testing, ,Advanced Informatics School (AIS), Universiti Teknologi Malaysia, International Campus, Jalan Semarak, Kuala Lumpur, Malaysia,
- Muller M. M. & Hagner O., Experiment about test-first programming. *IF.F.F, Proceedings-Software*, 149(5):131-136, 2002. Retrieved 8/30/2014 from <https://connect.spsu.edu/stamp/,DanaInfo=.aifgh1urvznJtqrsO48y+stamp.jsp?tp=&ar number=1049202>
- Murphy, J. & Stone, C. (1995, December). International Conference on Object Oriented Information Systems, 18-20 December 1995, Dublin, Ireland, Proceedings. Springer, 1996, ISBN 3-540-76010-5.
- Murphy, C., et al. (2009). Using JML runtime assertion checking to automate metamorphic testing in applications without test oracles. *Software Testing Verification and Validation, 2009. ICST '09. International Conference on*. Retrieved 8/30/2014 from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&ar number=4815377&is number=4815322>
- NUnit. NUnit.org, 2015. Retrieved 5/29/2015 from <http://www.nunit.org/>
- Pancur M., Ciglaric M., Trampus, M. and Vidmar T. (2003). Towards empirical evaluation of test-driven development in a university environment. *In Proceedings of FUROCON 2003. Computer as a Tool. The IFFF Region 8*, volume 2, pages 83-86, 2003. Retrieved 8/30/2014 from <https://connect.spsu.edu/stamp/,DanaInfo=.aifgh1urvznJtqrsO48y+stamp.jsp?tp=&ar number=1248153>
- Plat, N., Van Katwijk, J. & Toetenel, H. (1992). Application and benefits of formal methods in software development. *Software Engineering Journal* , vol.7, no.5, pp.335,346, Sep 1992. Retrieved 9/26/2014 from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&ar number=165489&is number=4260>
- Preserve Articles. (2011). Essay on the importance of Computer in the Modern Society. Retrieved 9/4/2014 from

<http://www.preservearticles.com/201103264739/importance-of-computer-in-the-modern-society.html>

Rutledge, R., Tsui, F., (2013). Formal specification-driven development. *Unpublished Student Dissertation. Southern Polytechnic State University, Marietta, GA* . Retrieved from [http://cse.spsu.edu/ftsui/images/Paper\\_FSDD%20SERP%202014%20Final.pdf](http://cse.spsu.edu/ftsui/images/Paper_FSDD%20SERP%202014%20Final.pdf)

Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M. & Erdogmus, H. (2014). What Do We Know about Test-Driven Development? *Software, IEEE* , vol.27, no.6, pp.16,19, Nov.-Dec. 2010. Retrieved 9/2/2014 from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5604358&isnumber=5604350>

Siniaalto, M. & Abrahamsson, P. (2007) A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage, *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, vol., no., pp.275,284, 20-21 Sept. 2007. Retrieved 8/30/2014 from [https://connect.spsu.edu/stamp/,DanaInfo=.aifgh1urvzn\]tqrsO48y+stamp.jsp?tp=&arnumber=4343755](https://connect.spsu.edu/stamp/,DanaInfo=.aifgh1urvzn]tqrsO48y+stamp.jsp?tp=&arnumber=4343755)

Solis, C. & W. Xiaofeng (2011). A study of the characteristics of behavior driven development. *Software Engineering and Advanced Applications (SEAA)*, 2011 37th EUROMICRO Conference on. Retrieved 9/25/2014 from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6068372&isnumber=6068309>

Sommerville, I. (2010). *Software Engineering*. Harlow, England: Addison-Wesley. ISBN: 978-0-13-703515-1

SpecFlow. SpecFlow - Cucumber for .NET, 2013. Retrieved 5/26/2015 from <http://specflow.org>

Staples, J. (1996). Do formal methods really work? *Australian Software Engineering Conference*. Proceedings of 1996. Retrieved 9/25/2014 from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=534124&isnumber=11149>

Tavares H.P., Guimarães G., Rezende, Mota V., Soares Manhães R., R., and Atem R. De Carvalho, A tool stack for implementing Behaviour-Driven Development in Python Language, CoRR, 2010. 10/25/2014 from <http://arxiv.org/pdf/1007.1722v1>

Vogel, L., & IDE, E. J. (2013). Eclipse IDE Tutorial. Retrieved 9/3/2014 from <http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&ved=0CFoQFjAI&url=http%3A%2F%2Fwww.cs.mun.ca%2F~mhatcher%2F3716%2FLecture%2520Slides%2FWeek%25206%2FEclipse%2520Quick%2520Guide.pdf&ei=H6QsVPLsDo3NggSvhoLwAw&usg=AFQjCNH1zBjCWPr1d55PO7yU2I3BzTUQDA&bvm=bv.76477589,d.eXY>

- Wedde, H.F., Cheng, B.H.C., Gries, D., Shankar, N., Lin, K.-J. & Ardis, M. (2014). Are formal methods useful for software development? *Computer Software and Applications Conference, 1992. COMPSAC '92. Proceedings.*, Sixteenth Annual International. Retrieved 9/25/2014 from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=217611&isnumber=5703>
- Wohlin C., Runeson P., Host M., Ohlsson M.C., Regnell B., and Wesslen A., Experimentation in Software Engineering: An Introduction. *Kluwer Academic*, 2000. Retrieved 10/10/2014 from <http://www.pdfbook.co.ke/details.php?title=Experimentation%20in%20Software%20Engineering&author=Martin%20H%F6st,%20Per%20Runeson&category=Computers&eid=115330&type=Book&popular=5>
- Wynne, M., Hellesoy, A. (2010). The Cucumber Book - Pragmatic Bookshelf. (n.d.). Retrieved 6/30/2015 from <http://media.pragprog.com/titles/hwcuc/gherkin.pdf>eck, *Test-Driven Development: By Example*. Boston: Addison-Wesley, 2003.

## 7. APPENDIX A FORMAL SPECIFICATION-DRIVEN DEVELOPMENT APPROACH METRICS

### 7.1. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT – ALL METRICS

These are all the metrics for the entire FSDD solution and extracted from the Visual Studio 2013 Code Analyzer. It includes the classes and methods

**Table 9: FSDD Metrics**

Scope	Type	Member	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Project			89	163	2	51	1203
Namespace			93	13	1	0	13
Type	Address		93	13	1	0	13
Member	Address	Address 0	100	1		0	1
Member	Address	Address 1.get() : string	98	1		0	1
Member	Address	Address 1.set(string) : void	95	1		0	1

Member	Address	Address 2.get() : string	98	1		0	1
Member	Address	Address 2.set(str ing) : void	95	1		0	1
Member	Address	City.get( ) : string	98	1		0	1
Member	Address	City.set( string) : void	95	1		0	1
Member	Address	Country .get() : string	98	1		0	1
Member	Address	Country .set(str ing) : void	95	1		0	1
Member	Address	State.ge t() : string	98	1		0	1
Member	Address	State.se t(string ) : void	95	1		0	1
Member	Address	Zip.get( ) : string	98	1		0	1
Member	Address	Zip.set( string) :	95	1		0	1

		void					
Namespace			94	31	3	18	62
Type	EventRepository		98	1	3	3	1
Member	EventRepository	EventRepository(DbContext)	98	1		3	1
Type	EventTypeRepository		98	1	3	3	1
Member	EventTypeRepository	EventTypeRepository(DbContext)	98	1		3	1
Type	GenericRepository<TDomain, TEntity>		75	25	1	12	55
Member	GenericRepository<TDomain, TEntity>	CheckModelState(TEntity) : void	81	2		3	2

Member	GenericRepository<TDomain, TEntity>	Convert(IEnumerable<TEntity>) : IEnumerable<TDomain>	77	2		3	3
Member	GenericRepository<TDomain, TEntity>	Convert(TEntity) : TDomain	100	1		0	0
Member	GenericRepository<TDomain, TEntity>	Delete(int) : TEntity	78	1		1	3
Member	GenericRepository<TDomain, TEntity>	Delete(TEntity) : TEntity	81	1		1	3
Member	GenericRepository<TDomain, TEntity>	GenericRepository(DbContext)	79	1		2	3



Member	GenericRepository<TDomain, TEntity>	Get(Expression<Func<TEntity, bool>>, Func<IQueryable<TEntity>, IOrderedQueryable<TEntity>>, string) : IEnumerable<TDomain>	61	4		9	10
Member	GenericRepository<TDomain, TEntity>	Get(int) : TDomain	73	2		1	5
Member	GenericRepository<TDomain, TEntity>	Get(TEntity) : TDomain	87	1		0	2
Member	GenericRepository<TDomain, TEntity>	GetAll() : IEnumerable<TDomain>	87	1		1	2
Member	GenericRepository<TDomain, TEntity>	GetAll(string) : IEnumerable<TDomain>	64	4		6	8

	TEntity>	Domain >					
Member	GenericRepository<TDomain, TEntity>	GetFirstOrDefault(Expression<Func<TEntity, bool>>, string) : TDomain	64	3		6	8
Member	GenericRepository<TDomain, TEntity>	Insert(TEntity) : TEntity	86	1		1	2
Member	GenericRepository<TDomain, TEntity>	Update(TEntity) : TEntity	75	1		3	4
Type	OrderRepository		98	1	3	3	1
Member	OrderRepository	OrderRepository(DbContext)	98	1		3	1
Type	PerformanceRepository		98	1	3	3	1

Member	PerformanceRepository	PerformanceRepository (DbContext)	98	1		3	1
Type	UnconvertedGenericRepository<TEntity>		96	2	2	2	3
Member	UnconvertedGenericRepository<TEntity>	Convert (TEntity): TEntity	91	1		0	2
Member	UnconvertedGenericRepository<TEntity>	UnconvertedGenericRepository (DbContext)	98	1		2	1
Namespace			93	149	2	16	156
Type	CardType		93	9	1	3	10
Member	CardType	CardType()	87	1		2	2
Member	CardType	CardType1.get(): string	98	1		0	1

Member	CardType	CardType1.set(string) : void	95	1		0	1
Member	CardType	CardType1.getId() : int	98	1		0	1
Member	CardType	CardType1.setId(int) : void	95	1		0	1
Member	CardType	Orders.getId() : ICollection<Order>	98	1		2	1
Member	CardType	Orders.set(ICollection<Order>) : void	95	1		2	1
Member	CardType	status.get() : bool	98	1		0	1
Member	CardType	status.set(bool) : void	95	1		0	1
Type	Event		92	25	1	7	26
Member	Event	CreateById.get()	98	1		1	1

		) : int?					
Member	Event	CreateById.setInt(int?) : void	95	1		1	1
Member	Event	Description.get() : string	98	1		0	1
Member	Event	Description.set(string) : void	95	1		0	1
Member	Event	Event()	87	1		2	2
Member	Event	EventId.get() : int	98	1		0	1
Member	Event	EventId.set(int) : void	95	1		0	1
Member	Event	EventType.get() : EventType	98	1		1	1
Member	Event	EventType.set(EventType) : void	95	1		1	1

Member	Event	EventTypeld.get() : int?	98	1		1	1
Member	Event	EventTypeld.set(int?) : void	95	1		1	1
Member	Event	Image.get() : string	98	1		0	1
Member	Event	Image.set(string) : void	95	1		0	1
Member	Event	Location.get() : string	98	1		0	1
Member	Event	Location.set(string) : void	95	1		0	1
Member	Event	Name.get() : string	98	1		0	1
Member	Event	Name.set(string) : void	95	1		0	1
Member	Event	Performances.get() : ICollection<Perf	98	1		2	1

		ormance>					
Member	Event	Performances.set(ICollection<Performance>) : void	95	1		2	1
Member	Event	Price.get() : decimal?	98	1		2	1
Member	Event	Price.set(decimal?) : void	95	1		2	1
Member	Event	status.get() : int	98	1		0	1
Member	Event	status.set(int) : void	95	1		0	1
Member	Event	User.get() : User	98	1		1	1
Member	Event	User.set(User) : void	95	1		1	1
Type	EventType		93	9	1	3	10

Member	EventTy pe	Events.g et() : ICollecti on<Eve nt>	98	1		2	1
Member	EventTy pe	Events.s et(IColl ection< Event>) : void	95	1		2	1
Member	EventTy pe	EventTy pe()	87	1		2	2
Member	EventTy pe	EventTy peld.get () : int	98	1		0	1
Member	EventTy pe	EventTy peld.set (int) : void	95	1		0	1
Member	EventTy pe	status.g et() : bool	98	1		0	1
Member	EventTy pe	status.s et(bool) : void	95	1		0	1
Member	EventTy pe	Type.ge t() : string	98	1		0	1
Member	EventTy pe	Type.se t(string	95	1		0	1



		) : void					
Type	ManagementToolProjectEntities		93	16	2	11	16
Member	ManagementToolProjectEntities	CardTypes.get(): DbSet<CardType>	98	1		2	1
Member	ManagementToolProjectEntities	CardTypes.set(DbSet<CardType>) : void	95	1		2	1
Member	ManagementToolProjectEntities	Events.get() : DbSet<Event>	98	1		2	1
Member	ManagementToolProjectEntities	Events.set(DbSet<Event>) : void	95	1		2	1
Member	ManagementToolProjectEntities	EventTypes.get(): DbSet<EventType>	98	1		2	1
Member	ManagementTool	EventTypes.set(	95	1		2	1

	IProjectEntities	DbSet<EventTy pe>): void					
Member	ManagementToolProjectEntities	ManagementToolProjectEntities()	98	1		1	1
Member	ManagementToolProjectEntities	OnModelCreating(DbModelBuilder): void	98	1		2	1
Member	ManagementToolProjectEntities	OrderPerformanceMapping2.get() : DbSet<OrderPerformanceMapping2>	98	1		2	1
Member	ManagementToolProjectEntities	OrderPerformanceMapping2.set(DbSet<OrderPerformanceMapping2>): void	95	1		2	1

Member	ManagementToolProjectEntities	Orders.get() : DbSet<Order>	98	1		2	1
Member	ManagementToolProjectEntities	Orders.set(DbSet<Order>) : void	95	1		2	1
Member	ManagementToolProjectEntities	Performances.get() : DbSet<Performance>	98	1		2	1
Member	ManagementToolProjectEntities	Performances.set(DbSet<Performance>) : void	95	1		2	1
Member	ManagementToolProjectEntities	Users.get() : DbSet<User>	98	1		2	1
Member	ManagementToolProjectEntities	Users.set(DbSet<User>) : void	95	1		2	1
Type	Order		92	41	1	7	42
Member	Order	BillingAddress.get() :	98	1		0	1

		string					
Member	Order	BillingAddress.set(string) : void	95	1		0	1
Member	Order	BillingCity.get() : string	98	1		0	1
Member	Order	BillingCity.set(string) : void	95	1		0	1
Member	Order	BillingState.get() : string	98	1		0	1
Member	Order	BillingState.set(string) : void	95	1		0	1
Member	Order	BillingZipCode.get() : int?	98	1		1	1
Member	Order	BillingZipCode.set(int?) : void	95	1		1	1
Member	Order	CardType.get() : CardTy	98	1		1	1

		pe					
Member	Order	CardType.set(CardType): void	95	1		1	1
Member	Order	CardType.getId(): int?	98	1		1	1
Member	Order	CardType.setId(int?): void	95	1		1	1
Member	Order	CreditCard.getId(): string	98	1		0	1
Member	Order	CreditCard.set(string): void	95	1		0	1
Member	Order	ExpirationDate.get(): DateTime?	98	1		2	1
Member	Order	ExpirationDate.set(DateTime?): void	95	1		2	1
Member	Order	FirstName.get()	98	1		0	1

		) : string					
Member	Order	FirstName.set(string) : void	95	1		0	1
Member	Order	LastName.get() : string	98	1		0	1
Member	Order	LastName.set(string) : void	95	1		0	1
Member	Order	Order()	87	1		2	2
Member	Order	OrderId.get() : int	98	1		0	1
Member	Order	OrderId.set(int) : void	95	1		0	1
Member	Order	OrderPerformanceMapping2.get() : ICollection<OrderPerformanceMapping2>	98	1		2	1

Member	Order	OrderPerformanceMapping2.set (ICollection<OrderPerformanceMapping2>) : void	95	1		2	1
Member	Order	Performance.get () : Performance	98	1		1	1
Member	Order	Performance.set (Performance) : void	95	1		1	1
Member	Order	PerformanceId.get() : int?	98	1		1	1
Member	Order	PerformanceId.set(int?) : void	95	1		1	1
Member	Order	ShippingAdress.get() : string	98	1		0	1

Member	Order	ShippingAddress.set(string) : void	95	1		0	1
Member	Order	ShippingCity.get() : string	98	1		0	1
Member	Order	ShippingCity.set(string) : void	95	1		0	1
Member	Order	ShippingState.get() : string	98	1		0	1
Member	Order	ShippingState.set(string) : void	95	1		0	1
Member	Order	ShippingZipCode.get() : int?	98	1		1	1
Member	Order	ShippingZipCode.set(int?) : void	95	1		1	1
Member	Order	status.get() : int	98	1		0	1



Member	Order	status.set(int) : void	95	1		0	1
Member	Order	TicketNumber.get() : int?	98	1		1	1
Member	Order	TicketNumber.set(int?) : void	95	1		1	1
Type	OrderPerformanceMapping2		93	13	1	3	13
Member	OrderPerformanceMapping2	ID.get() : int	98	1		0	1
Member	OrderPerformanceMapping2	ID.set(int) : void	95	1		0	1
Member	OrderPerformanceMapping2	OderId.get() : int	98	1		0	1
Member	OrderPerformanceMapping2	OderId.set(int) : void	95	1		0	1

Member	OrderPerformanceMapping2	Order.get() : Order	98	1		1	1
Member	OrderPerformanceMapping2	Order.set(Order) : void	95	1		1	1
Member	OrderPerformanceMapping2	OrderPerformanceMapping2()	100	1		0	1
Member	OrderPerformanceMapping2	Performance.get() : Performance	98	1		1	1
Member	OrderPerformanceMapping2	Performance.set(Performance) : void	95	1		1	1
Member	OrderPerformanceMapping2	PerformanceId.get() : int	98	1		0	1
Member	OrderPerformanceMapping2	PerformanceId.set(int) : void	95	1		0	1
Member	OrderPerformance	Quantity.get() :	98	1		1	1

	ceMapping2	int?					
Member	OrderPerformanceMapping2	Quantity.set(int?): void	95	1		1	1
Type	Performance		92	21	1	8	23
Member	Performance	Date.get(): DateTime?	98	1		2	1
Member	Performance	Date.set(DateTime?): void	95	1		2	1
Member	Performance	Event.get(): Event	98	1		1	1
Member	Performance	Event.set(Event): void	95	1		1	1
Member	Performance	EventId.get(): int?	98	1		1	1
Member	Performance	EventId.set(int?): void	95	1		1	1
Member	Performance	EventName.get	98	1		0	1

	ance	() : string					
Member	Perform ance	EventN ame.set (string) : void	95	1		0	1
Member	Perform ance	OrderPe rforman ceMapp ing2.get () : ICollecti on<Ord erPerfo rmance Mappin g2>	98	1		2	1
Member	Perform ance	OrderPe rforman ceMapp ing2.set (I Collec tion<Or derPerf ormanc eMappi ng2>) : void	95	1		2	1
Member	Perform ance	Orders. get() : ICollecti on<Ord er>	98	1		2	1
Member	Perform ance	Orders.s et(IColl ection< Order>)	95	1		2	1

		: void					
Member	Performance	Performance() Performance()	80	1		3	3
Member	Performance	Performance.getId() getId() : int	98	1		0	1
Member	Performance	Performance.setId(int) setId(int) : void	95	1		0	1
Member	Performance	Price.getPrice() get() : decimal?	98	1		2	1
Member	Performance	Price.setPrice(decimal?) set(decimal?) : void	95	1		2	1
Member	Performance	status.getStatus() getStatus() : bool	98	1		0	1
Member	Performance	status.setStatus(bool) setStatus(bool) : void	95	1		0	1
Member	Performance	TotalTickets.getTotalTickets() getTotalTickets() : int?	98	1		1	1
Member	Performance	TotalTickets.getTotalTickets() TotalTickets	95	1		1	1

	ance	kets.set (int?): void					
Type	User		93	15	1	3	16
Member	User	EmailA ddress.g et(): string	98	1		0	1
Member	User	EmailA ddress.s et(strin g): void	95	1		0	1
Member	User	Events.g et(): ICollecti on<Eve nt>	98	1		2	1
Member	User	Events.s et(IColl ection< Event> ): void	95	1		2	1
Member	User	FirstNa me.get( ): string	98	1		0	1
Member	User	FirstNa me.set(s tring): void	95	1		0	1
Member	User	LastNa me.get(	98	1		0	1

		) : string					
Member	User	LastName.set(string) : void	95	1		0	1
Member	User	Password.get() : string	98	1		0	1
Member	User	Password.set(string) : void	95	1		0	1
Member	User	status.get() : bool	98	1		0	1
Member	User	status.set(bool) : void	95	1		0	1
Member	User	User()	87	1		2	2
Member	User	UserId.get() : int	98	1		0	1
Member	User	UserId.set(int) : void	95	1		0	1
Namespace			58	55	1	51	240
Type	Account		56	8	1	15	44

	Mediator						
Member	Account Mediator	Account Mediator()	100	1		0	1
Member	Account Mediator	Authenticate(RegisterVM): bool	52	4		13	21
Member	Account Mediator	RegisterUser(RegisterVM): bool	51	3		14	22
Type	OrderMediator		57	12	1	16	35
Member	OrderMediator	AddPerformanceOrder(int): void	56	4		11	13
Member	OrderMediator	CreateOrder(PaymentVM): bool	49	7		15	21
Member	OrderMediator	OrderMediator()	100	1		0	1
Type	TicketMe		62	35	1	42	161



	diator						
Member	TicketMediator	ClearCart() : void	91	1		2	1
Member	TicketMediator	CompleteOrder(CartVM) : void	100	1		1	0
Member	TicketMediator	CreateEvent(EventVM) : bool	62	2		7	8
Member	TicketMediator	CreatePerformance(PerformanceVM) : bool	61	2		9	9
Member	TicketMediator	GetActiveEvents() : List<EventVM>	62	2		15	9
Member	TicketMediator	GetAllPerformances() : List<PerformanceVM>	60	2		18	9
Member	TicketMediator	GetCart() : CartVM	67	2		3	7

Member	TicketMediator	GetEvent(int) : EventVM	58	2		15	12
Member	TicketMediator	GetEventItems() : List<EventItem>	65	2		5	7
Member	TicketMediator	GetEvents() : List<EventVM>	64	2		9	9
Member	TicketMediator	GetEventTypes() : List<Category>	63	2		9	9
Member	TicketMediator	GetPerformance(int) : PerformanceVM	58	2		15	12
Member	TicketMediator	GetPerformances(int) : List<PerformanceVM>	57	2		18	13
Member	TicketMediator	GetUpcomingPerformances() :	58	4		19	10

		List<PerformanceVM>					
Member	TicketMediator	HasEnoughSeats(int, int) : bool	70	2		1	6
Member	TicketMediator	TicketMediator()	100	1		0	1
Member	TicketMediator	UpdateEvent(EventVM) : bool	53	2		16	19
Member	TicketMediator	UpdatePerformance(PerformanceVM) : bool	52	2		17	20
Namespace			93	13	1	0	13
Type	RegisterVM		93	13	1	0	13
Member	RegisterVM	Email.get() : string	98	1		0	1
Member	RegisterVM	Email.set(string	95	1		0	1

		) : void					
Member	Register VM	FirstName.get() : string	98	1		0	1
Member	Register VM	FirstName.set(string) : void	95	1		0	1
Member	Register VM	Id.get() : int	98	1		0	1
Member	Register VM	Id.set(int) : void	95	1		0	1
Member	Register VM	LastName.get() : string	98	1		0	1
Member	Register VM	LastName.set(string) : void	95	1		0	1
Member	Register VM	Password.get() : string	98	1		0	1
Member	Register VM	Password.set(string) : void	95	1		0	1
Member	Register VM	Register VM()	100	1		0	1

Member	Register VM	VerifyPassword.get() : string	98	1		0	1
Member	Register VM	VerifyPassword.set(string) : void	95	1		0	1
Namespace			95	31	1	4	32
Type	CartVM		93	5	1	3	6
Member	CartVM	CartVM()	87	1		2	2
Member	CartVM	Performances.get() : List<PerformanceVM>	98	1		2	1
Member	CartVM	Performances.set(List<PerformanceVM>) : void	95	1		2	1
Member	CartVM	Total.get() : decimal	98	1		1	1

Member	CartVM	Total.set(decimal): void	95	1		1	1
Type	ConfirmationVM		100	1	1	0	1
Member	ConfirmationVM	ConfirmationVM()	100	1		0	1
Type	PaymentVM		92	25	1	1	25
Member	PaymentVM	BillingAddress.get(): Address	98	1		1	1
Member	PaymentVM	BillingAddress.set(Address): void	95	1		1	1
Member	PaymentVM	CreditCardNumber.get(): string	98	1		0	1
Member	PaymentVM	CreditCardNumber.set(string): void	95	1		0	1

Member	Payment VM	CreditCardType. get() : string	98	1		0	1
Member	Payment VM	CreditCardType. .set(string) : void	95	1		0	1
Member	Payment VM	CVV.get() (): string	98	1		0	1
Member	Payment VM	CVV.set( string) : void	95	1		0	1
Member	Payment VM	Email.get() (): string	98	1		0	1
Member	Payment VM	Email.set( string) : void	95	1		0	1
Member	Payment VM	ExpirationMonth. get() : string	98	1		0	1
Member	Payment VM	ExpirationMonth. set(string) : void	95	1		0	1

Member	Payment VM	ExpirationYear.get() : string	98	1		0	1
Member	Payment VM	ExpirationYear.set(string) : void	95	1		0	1
Member	Payment VM	FirstName.get() : string	98	1		0	1
Member	Payment VM	FirstName.set(string) : void	95	1		0	1
Member	Payment VM	LastName.get() : string	98	1		0	1
Member	Payment VM	LastName.set(string) : void	95	1		0	1
Member	Payment VM	NumberOfCards.get() : string	98	1		0	1
Member	Payment VM	NumberOfCards.set(string) : void	95	1		0	1



Member	Payment VM	PaymentVM()	100	1		0	1
Member	Payment VM	SameAsBilling.get() : bool	98	1		0	1
Member	Payment VM	SameAsBilling.set(bool) : void	95	1		0	1
Member	Payment VM	ShippingAddress.get() : Address	98	1		1	1
Member	Payment VM	ShippingAddress.set(Address) : void	95	1		1	1
Namespace			94	14	1	0	14
Type	Category		94	7	1	0	7
Member	Category	Category()	100	1		0	1
Member	Category	CategoryId.get() : int	98	1		0	1

Member	Category	CategoryId.set(int) : void	95	1		0	1
Member	Category	Name.get() : string	98	1		0	1
Member	Category	Name.set(string) : void	95	1		0	1
Member	Category	SelectedIndicator.get() : bool	98	1		0	1
Member	Category	SelectedIndicator.set(bool) : void	95	1		0	1
Type	EventItem		94	7	1	0	7
Member	EventItem	EventId.get() : int	98	1		0	1
Member	EventItem	EventId.set(int) : void	95	1		0	1
Member	EventItem	EventItem()	100	1		0	1

Member	EventItem	Name.get() : string	98	1		0	1
Member	EventItem	Name.set(string) : void	95	1		0	1
Member	EventItem	SelectedIndicator.get() : bool	98	1		0	1
Member	EventItem	SelectedIndicator.set(bool) : void	95	1		0	1
Namespace			93	57	1	6	57
Type	BuyTicketsVM		93	15	1	4	15
Member	BuyTicketsVM	BuyTicketsVM()	100	1		0	1
Member	BuyTicketsVM	Categories.get() : List<Category>	98	1		2	1
Member	BuyTicketsVM	Categories.set(List<Category>) :	95	1		2	1

		void					
Member	BuyTicketsVM	Events.get() : List<EventItem>	98	1		2	1
Member	BuyTicketsVM	Events.set(List<EventItem>) : void	95	1		2	1
Member	BuyTicketsVM	FromDate.get() : string	98	1		0	1
Member	BuyTicketsVM	FromDate.set(string) : void	95	1		0	1
Member	BuyTicketsVM	LblFromDate.get() : string	98	1		0	1
Member	BuyTicketsVM	LblFromDate.set(string) : void	95	1		0	1
Member	BuyTicketsVM	LblToDate.get() : string	98	1		0	1

Member	BuyTicketsVM	LblToDate.set(string) : void	95	1		0	1
Member	BuyTicketsVM	Performances.get() : List<PerformanceVM>	98	1		2	1
Member	BuyTicketsVM	Performances.set(List<PerformanceVM>) : void	95	1		2	1
Member	BuyTicketsVM	ToDate.get() : string	98	1		0	1
Member	BuyTicketsVM	ToDate.set(string) : void	95	1		0	1
Type	EventVM		93	21	1	3	21
Member	EventVM	Active.get() : bool	98	1		0	1
Member	EventVM	Active.set(bool) : void	95	1		0	1

Member	EventVM	Categories.get() : List<Category>	98	1		2	1
Member	EventVM	Categories.set(List<Category>) : void	95	1		2	1
Member	EventVM	Category.get() : int	98	1		0	1
Member	EventVM	Category.set(int) : void	95	1		0	1
Member	EventVM	CategoryName.get() : string	98	1		0	1
Member	EventVM	CategoryName.set(string) : void	95	1		0	1
Member	EventVM	Description.get() : string	98	1		0	1
Member	EventVM	Description.set(string) :	95	1		0	1

		void					
Member	EventVM	EventVM()	100	1		0	1
Member	EventVM	Id.get(): int	98	1		0	1
Member	EventVM	Id.set(int): void	95	1		0	1
Member	EventVM	Image.get(): string	98	1		0	1
Member	EventVM	Image.set(string): void	95	1		0	1
Member	EventVM	Location.get(): string	98	1		0	1
Member	EventVM	Location.set(string): void	95	1		0	1
Member	EventVM	Name.get(): string	98	1		0	1
Member	EventVM	Name.set(string): void	95	1		0	1

Member	EventVM	Performances.get() : List<PerformanceVM>	98	1		2	1
Member	EventVM	Performances.set(List<PerformanceVM>) : void	95	1		2	1
Type	PerformanceVM		93	21	1	2	21
Member	PerformanceVM	AvailableTickets.get() : int	98	1		0	1
Member	PerformanceVM	AvailableTickets.set(int) : void	95	1		0	1
Member	PerformanceVM	Cancelled.get() : bool	98	1		0	1
Member	PerformanceVM	Cancelled.set(bool) : void	95	1		0	1
Member	PerformanceVM	EventId.get() : int	98	1		0	1



Member	PerformanceVM	EventId.set(int) : void	95	1		0	1
Member	PerformanceVM	EventName.get() : string	98	1		0	1
Member	PerformanceVM	EventName.set(string) : void	95	1		0	1
Member	PerformanceVM	LineNumber.get() : int	98	1		0	1
Member	PerformanceVM	LineNumber.set(int) : void	95	1		0	1
Member	PerformanceVM	Location.get() : string	98	1		0	1
Member	PerformanceVM	Location.set(string) : void	95	1		0	1
Member	PerformanceVM	PerformanceDate.get() : string	98	1		0	1

Member	PerformanceVM	PerformanceDate.set(string) : void	95	1		0	1
Member	PerformanceVM	PerformanceId.get() : int	98	1		0	1
Member	PerformanceVM	PerformanceId.set(int) : void	95	1		0	1
Member	PerformanceVM	PerformanceVM()	100	1		0	1
Member	PerformanceVM	Price.get() : decimal	98	1		1	1
Member	PerformanceVM	Price.set(decimal) : void	95	1		1	1
Member	PerformanceVM	Quantity.get() : int	98	1		0	1
Member	PerformanceVM	Quantity.set(int) : void	95	1		0	1
Namespace			84	24	1	20	35

Type	Account Transformer		100	2	1	1	1
Member	Account Transformer	Account Transformer()	100	1		0	1
Member	Account Transformer	Transform(RegisterVM) : void	100	1		1	0
Type	Category Transformer		77	5	1	7	9
Member	Category Transformer	CategoryTransformer()	100	1		0	1
Member	Category Transformer	Transform(EventType) : Category	77	1		2	3
Member	Category Transformer	Transform(IEnumerable<EventType>) : List<Category>	71	3		7	5
Type	EventTransformer		70	7	1	13	13

Member	EventTransformer	EventTransformer()	100	1		0	1
Member	EventTransformer	Transform(Event): EventVM	62	3		10	7
Member	EventTransformer	Transform(IEnumerable<Event>): List<EventVM>	71	3		7	5
Type	OrderTransformer		100	1	1	0	1
Member	OrderTransformer	OrderTransformer()	100	1		0	1
Type	PerformanceTransformer		73	9	1	13	11
Member	PerformanceTransformer	PerformanceTransformer()	100	1		0	1
Member	PerformanceTransformer	Transform(IEnumerable<Perfo	64	7		11	7

		rmance >): List<Pe rforman ceVM>					
Member	Perform anceTra nsformer	Transfo rm(Perf ormanc e): Perform anceVM	72	1		6	3

Scope	Approach	Namespace	Maintainability	Cyclomatic	Depth of	Class Coupli	Lines of
-------	----------	-----------	-----------------	------------	----------	--------------	----------

			Index	Complexity	Inheritance	ng	Code
Method	FSDD	Ticketing.Framework.Models.Cart	100	1		0	1
Method	FSDD	Ticketing.Framework.BusinessModels	93	13	1	0	13
Method	FSDD	Ticketing.Framework.Data	98	1	3	3	1
Method	FSDD	Ticketing.Framework.Data	98	1	3	3	1
Method	FSDD	Ticketing.Framework.Data	75	25	1	12	55
Method	FSDD	Ticketing.Framework.Data	98	1	3	3	1
Method	FSDD	Ticketing.Framework.Data	98	1	3	3	1
Method	FSDD	Ticketing.Framework.Data	96	2	2	2	3
Method	FSDD	Ticketing.Framework.DBModels	93	9	1	3	10
Method	FSDD	Ticketing.Framework.DBModels	92	25	1	7	26
Method	FSDD	Ticketing.Framework.DBModels	93	9	1	3	10

Method	FSDD	Ticketing.Framework. DBModels	93	16	2	11	16
Method	FSDD	Ticketing.Framework. DBModels	92	41	1	7	42
Method	FSDD	Ticketing.Framework. DBModels	93	13	1	3	13
Method	FSDD	Ticketing.Framework. DBModels	92	21	1	8	23
Method	FSDD	Ticketing.Framework. DBModels	93	15	1	3	16
Method	FSDD	Ticketing.Framework. Mediators	56	8	1	15	44
Method	FSDD	Ticketing.Framework. Mediators	57	12	1	16	35
Method	FSDD	Ticketing.Framework. Mediators	62	35	1	42	161
Method	FSDD	Ticketing.Framework. Models.Account	93	13	1	0	13
Method	FSDD	Ticketing.Framework. Models.Cart	93	5	1	3	6
Method	FSDD	Ticketing.Framework. Models.Cart	100	1	1	0	1
Method	FSDD	Ticketing.Framework. Models.Cart	92	25	1	1	25



Method	FSDD	Ticketing.Framework. Models.Common	94	7	1	0	7
Method	FSDD	Ticketing.Framework. Models.Common	94	7	1	0	7
Method	FSDD	Ticketing.Framework. Models.Ticket	93	15	1	4	15
Method	FSDD	Ticketing.Framework. Models.Ticket	93	21	1	3	21
Method	FSDD	Ticketing.Framework. Models.Ticket	93	21	1	2	21
Method	FSDD	Ticketing.Framework. Transformers	100	2	1	1	1
Method	FSDD	Ticketing.Framework. Transformers	77	5	1	7	9
Method	FSDD	Ticketing.Framework. Transformers	70	7	1	13	13
Method	FSDD	Ticketing.Framework. Transformers	100	1	1	0	1
Method	FSDD	Ticketing.Framework. Transformers	73	9	1	13	11
Method	FSDD	CheckOutSteps()	100	1		0	1
Method	FSDD	GivenThatThereAreIt emsInTheShoppingCa rt() : void	77	1		4	2

Method	FSDD	ThenTheSystemDisplaysOrderConfirmation() : void	84	1		3	2
Method	FSDD	ThenTheSystemDoesNotUpdateInventory() : void	94	1		2	1
Method	FSDD	ThenTheSystemUpdatesTheInventory() : void	77	1		4	2
Method	FSDD	WhenTheUserCancelsTheOrder() : void	81	1		3	1
Method	FSDD	WhenTheUserConfirmsTheOrder() : void	74	1		4	2
Method	FSDD	Main(string[]) : void	100	1		0	0
Method	FSDD	Program()	100	1		0	1

## 7.2. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT CLASS-LEVEL METRICS

These are all the class-level metrics for the FSDD solution created from the main spreadsheet of all the metrics.

**Table 10: FSDD Class-Level Metrics**

Scope	Approach	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Class	FSDD	93	13	1	0	13
Class	FSDD	94	31	3	18	62
Class	FSDD	93	149	2	16	156
Class	FSDD	58	55	1	51	240
Class	FSDD	93	13	1	0	13
Class	FSDD	95	31	1	4	32
Class	FSDD	94	14	1	0	14
Class	FSDD	93	57	1	6	57
Class	FSDD	84	24	1	20	35
Class	FSDD	100	2	1	0	1

Class	FSDD	83	7	1	13	11
-------	------	----	---	---	----	----

### 7.3. FORMAL SPECIFICATION-DRIVEN DEVELOPMENT PROJECT-LEVEL METRICS

These are the metrics for the FSDD solution created from the main spreadsheet of the FSDD metrics.

**Table 11: FSDD Project-Level Metrics**

Scope	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Project	89	163	2	51	1203

## 8. APPENDIX B SOFTWARE DEVELOPMENT LIFE CYCLE APPROACH METRICS

### 8.1. SOFTWARE DEVELOPMENT LIFE CYCLE - ALL METRICS

These are all the metrics for the FSDD solution extracted from Visual Studio Code Analyzer output.

Table 12: SDLC Metrics

Scope	Type	Member	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Project			80	833	5	210	2405
Type	NVPAPICaller		57	27	4	15	111
Member	NVPAPICaller	buildCredentialsNVPString() : string	60	5		2	12
Member	NVPAPICaller	DoCheckoutPayment(string, string, string, ref NVPCodec, ref string) : bool	52	4		2	19

Member	NVPAPICaller	GetCheckoutDetails(string, ref string, ref NVPCodec, ref string) : bool	54	4		2	16
Member	NVPAPICaller	HttpCall(string) : string	55	4		8	16
Member	NVPAPICaller	IsEmpty(string) : bool	83	2		0	2
Member	NVPAPICaller	NVPAPICaller()	63	1		1	10
Member	NVPAPICaller	NVPAPICaller()	89	1		0	1
Member	NVPAPICaller	SetCredentials(string, string, string) : void	80	1		0	3
Member	NVPAPICaller	ShortcutExpressCheckout(string, ref string, ref string) : bool	45	5		7	32
Type	NVPCodec		76	14	3	5	32
Member	NVPCodec	Add(string, string, int) : void	92	1		1	1
Member	NVPCodec	Decode(string) : void	65	3		2	8

Member	NVPCodec	Encode() : string	60	3		3	12
Member	NVPCodec	GetArrayName (int, string) : string	75	2		1	4
Member	NVPCodec	NVPCodec()	100	1		1	1
Member	NVPCodec	NVPCodec()	83	1		0	2
Member	NVPCodec	Remove(string, int) : void	92	1		1	1
Member	NVPCodec	this.get(string, int) : string	84	1		1	2
Member	NVPCodec	this.set(string, int, string) : void	92	1		1	1
Namesp ace			84	36	5	48	82
Type	BundleCon fig		75	2	1	5	6
Member	BundleCon fig	BundleConfig()	100	1		0	1
Member	BundleCon fig	RegisterBundl es(BundleCollec tion) : void	68	1		5	5
Type	FilterConfig		95	2	1	2	2

Member	FilterConfig	FilterConfig()	100	1		0	1
Member	FilterConfig	RegisterGlobalFilters(GlobalFilterCollection) : void	94	1		2	1
Type	MvcApplication		84	2	2	8	5
Member	MvcApplication	Application_Start() : void	76	1		7	4
Member	MvcApplication	MvcApplication()	100	1		1	1
Type	RouteConfig		90	2	1	4	3
Member	RouteConfig	RegisterRoutes(RouteCollection) : void	82	1		4	2
Member	RouteConfig	RouteConfig()	100	1		0	1
Type	Site		62	26	5	28	65
Member	Site	FillPage() : void	53	8		8	15
Member	Site	ImageButton1_Click(object, ImageClickEventArgs) : void	94	1		3	1



Member	Site	lbRegister_Click(object, EventArgs) : void	94	1		3	1
Member	Site	lbSignOut_Click(object, EventArgs) : void	51	5		11	21
Member	Site	LoadPage() : void	49	8		11	22
Member	Site	Page_Load(object, EventArgs) : void	91	1		1	2
Member	Site	Search_Click(object, EventArgs) : void	80	1		5	2
Member	Site	Site()	100	1		1	1
Type	TermofService		100	2	4	2	1
Member	TermofService	Page_Load(object, EventArgs) : void	100	1		1	0
Member	TermofService	TermofService()	100	1		1	1
Namespace			58	32	4	25	122

Type	ManageAccount		58	32	4	25	122
Member	ManageAccount	AccEdit_Click(object, EventArgs) : void	81	1		3	3
Member	ManageAccount	EditAccCancel_Click(object, EventArgs) : void	73	1		3	5
Member	ManageAccount	EditAccSave_Click(object, EventArgs) : void	49	7		16	24
Member	ManageAccount	EditLogCancel_Click(object, EventArgs) : void	73	1		3	5
Member	ManageAccount	EditLogSave_Click(object, EventArgs) : void	48	11		13	25
Member	ManageAccount	IsValidEmail(string) : bool	75	2		1	5
Member	ManageAccount	LoginEdit_Click(object, EventArgs) : void	81	1		3	3
Member	ManageAccount	ManageAccount	100	1		1	1

	count	nt()					
Member	ManageAccount	Page_Load(object, EventArgs) : void	71	2		4	6
Member	ManageAccount	RefreshAccountInfo() : void	40	5		14	45
Namespace			75	129	5	81	397
Type	AdministratorPage		100	2	4	2	1
Member	AdministratorPage	AdministratorPage()	100	1		1	1
Member	AdministratorPage	Page_Load(object, EventArgs) : void	100	1		1	0
Type	AdminSite		73	15	5	14	41
Member	AdminSite	Admin_Click(object, EventArgs) : void	94	1		3	1
Member	AdminSite	AdminSite()	100	1		1	1
Member	AdminSite	Cr_Cat_Click(object, EventArgs) : void	94	1		3	1

Member	AdminSite	Cr_Event_Click (object, EventArgs) : void	94	1		3	1
Member	AdminSite	Cr_PType_Clic k(object, EventArgs) : void	94	1		3	1
Member	AdminSite	Cr_User_Click( object, EventArgs) : void	94	1		3	1
Member	AdminSite	Cr_Venue_Clic k(object, EventArgs) : void	94	1		3	1
Member	AdminSite	lbSignOut_Clic k(object, EventArgs) : void	53	3		10	19
Member	AdminSite	LoadPage() : void	57	4		7	14
Member	AdminSite	Page_Load(obj ect, EventArgs) : void	100	1		1	1
Type	CreateE_Ca tegrory		69	9	4	22	28
Member	CreateE_Ca tegrory	CateGrid_Row Command(obj	63	2		11	8

		ect, GridViewCom mandEventArg s) : void					
Member	CreateE_Ca tegrory	CateGrid_Row Updating(obje ct, GridViewUpda teEventArgs) : void	95	1		2	1
Member	CreateE_Ca tegrory	CreateE_Categ ory()	100	1		1	1
Member	CreateE_Ca tegrory	Page_Load(obj ect, EventArgs) : void	86	2		4	2
Member	CreateE_Ca tegrory	SubCatBut_Clic k(object, EventArgs) : void	56	3		10	16
Type	CreateEven t		61	46	4	61	147
Member	CreateEven t	buttonSave_Cli ck(object, EventArgs) : void	36	13		31	59
Member	CreateEven t	ClearTextFields () : void	70	1		2	6
Member	CreateEven t	CreateEvent()	100	1		1	1

Member	CreateEvent	CustomValidator_ServerValidate(object, ServerValidateEventArgs) : void	88	2		2	1
Member	CreateEvent	CustomValidator1_ServerValidate(object, ServerValidateEventArgs) : void	86	2		3	1
Member	CreateEvent	CustomValidator2_ServerValidate(object, ServerValidateEventArgs) : void	88	2		2	1
Member	CreateEvent	CustomValidator3_ServerValidate(object, ServerValidateEventArgs) : void	86	2		3	1
Member	CreateEvent	eDetails_Item Deleted(object , DetailsViewDeletedEventArgs) : void	81	1		3	3
Member	CreateEvent	eDetails_Item Updated(object, DetailsViewUp	98	1		2	1

		datedEventArgs) : void					
Member	CreateEvent	EventGrid_RowDeleting(object, GridViewDeleteEventArgs) : void	61	2		12	10
Member	CreateEvent	GridView1_SelectedIndexChanged(object, GridViewSelectEventArgs) : void	69	2		10	5
Member	CreateEvent	isImage(string) : bool	71	5		1	4
Member	CreateEvent	ListBox_SelectedIndexChanged(object, EventArgs) : void	91	1		5	1
Member	CreateEvent	Page_Load(object, EventArgs) : void	100	1		1	1
Member	CreateEvent	price_Click(object, EventArgs) : void	41	8		26	46
Member	CreateEvent	ShowImages() : void	67	2		4	6

Type	CreateP_Method		71	6	4	13	16
Member	CreateP_Method	C_PTButt_Click(object, EventArgs) : void	60	3		8	12
Member	CreateP_Method	CreateP_Method()	100	1		1	1
Member	CreateP_Method	Page_Load(object, EventArgs) : void	80	2		4	3
Type	CreateVenue		57	42	4	54	152
Member	CreateVenue	AddSeat_Click(object, EventArgs) : void	42	9		23	43
Member	CreateVenue	ClearVenueFields() : void	63	1		4	10
Member	CreateVenue	CreateVenue()	100	1		1	1
Member	CreateVenue	CustomValidator_ServerValidate(object, ServerValidateEventArgs) : void	86	3		1	1



Member	CreateVenue	GridView1_SelectedIndexChanging(object, GridViewSelectEventArgs) : void	69	2		10	5
Member	CreateVenue	isImage(string) : bool	71	5		1	4
Member	CreateVenue	Page_Load(object, EventArgs) : void	75	2		6	4
Member	CreateVenue	TextValidate(object, ServerValidateEventArgs) : void	90	1		2	1
Member	CreateVenue	VenDetail_ItemDeleted(object, DetailsViewDeletedEventArgs) : void	81	1		3	3
Member	CreateVenue	VenDetail_ItemUpdated(object, DetailsViewUpdatedEventArgs) : void	86	1		4	2
Member	CreateVenue	VenueGrid_RowDeleting(object, GridViewDeleteEventArgs) :	61	2		13	10

		void					
Member	CreateVenue	venueSave_Click(object, EventArgs) : void	35	14		32	68
Type	Seatlevel		92	9	1	0	12
Member	Seatlevel	ID.get() : int	98	1		0	1
Member	Seatlevel	ID.set(int) : void	95	1		0	1
Member	Seatlevel	Seatlevel(int, int, int)	78	1		0	4
Member	Seatlevel	SeatTotal.get() : int	98	1		0	1
Member	Seatlevel	SeatTotal.set(int) : void	95	1		0	1
Member	Seatlevel	Section.get() : int	98	1		0	1
Member	Seatlevel	Section.set(int) : void	95	1		0	1
Member	Seatlevel	VID.get() : int	98	1		0	1
Member	Seatlevel	VID.set(int) : void	95	1		0	1
Namesp			88	255	1	35	825

ace							
Type	Cart		84	5	1	1	12
Member	Cart	Cart()	76	1		1	4
Member	Cart	Cart(string, List<int>)	76	1		1	4
Member	Cart	Cusname.get() : string	98	1		0	1
Member	Cart	Cusname.set(s tring) : void	95	1		0	1
Member	Cart	getList() : List<int>	91	1		1	2
Type	CartItem		93	17	1	3	17
Member	CartItem	CartItem()	100	1		0	1
Member	CartItem	catenum.get() : int	98	1		0	1
Member	CartItem	catenum.set(in t) : void	95	1		0	1
Member	CartItem	EventDate.get( ) : DateTime	98	1		1	1
Member	CartItem	EventDate.set( DateTime) : void	95	1		1	1

Member	CartItem	EventName.get() : string	98	1		0	1
Member	CartItem	EventName.set(string) : void	95	1		0	1
Member	CartItem	eventtime.get() : TimeSpan	98	1		1	1
Member	CartItem	eventtime.set(TimeSpan) : void	95	1		1	1
Member	CartItem	Price.get() : decimal	98	1		1	1
Member	CartItem	Price.set(decimal) : void	95	1		1	1
Member	CartItem	seatid.get() : int	98	1		0	1
Member	CartItem	seatid.set(int) : void	95	1		0	1
Member	CartItem	seatnum.get() : int	98	1		0	1
Member	CartItem	seatnum.set(int) : void	95	1		0	1
Member	CartItem	VenueName.get() : string	98	1		0	1
Member	CartItem	VenueName.se	95	1		0	1

		t(string) : void					
Type	Category		92	7	1	0	10
Member	Category	cat.get() : string	98	1		0	1
Member	Category	cat.set(string) : void	95	1		0	1
Member	Category	Category(int, string, string)	78	1		0	4
Member	Category	ID.get() : int	98	1		0	1
Member	Category	ID.set(int) : void	95	1		0	1
Member	Category	subcat.get() : string	98	1		0	1
Member	Category	subcat.set(stri ng) : void	95	1		0	1
Type	Connection		57	109	1	32	613
Member	Connection	AddEvents(Eve nt) : int	56	3		7	14
Member	Connection	AddSeat(int, double, int) : void	64	2		5	10
Member	Connection	AddSeatCat(int , int) : int	61	2		4	12

Member	Connection	AddSeatSection(Seatlevel) : void	65	1		4	8
Member	Connection	AddSubCat(string, string) : int	59	3		5	14
Member	Connection	AddVenue(Venue) : int	56	3		7	14
Member	Connection	ChangeSeatStatus(int, string) : void	65	2		4	9
Member	Connection	CheckUserExist(string) : int	59	3		5	14
Member	Connection	Connection()	68	1		4	6
Member	Connection	CreateOrder(int, DateTime, double, string) : int	58	3		6	14
Member	Connection	CreatePtype(string) : int	59	3		5	14
Member	Connection	CreateTicket(int, int, double) : void	65	2		5	9
Member	Connection	CusRegister(int, string, string, string, string, string, string, string,	57	3		5	14

		string, int) : int					
Member	Connection	Decrypt(string) : string	58	1		10	12
Member	Connection	Encrypt(string) : string	55	3		10	15
Member	Connection	GetAvlSeat(int, int, string, string) : DataSet	60	2		6	12
Member	Connection	GetCartinfo(int ) : CartItem	52	3		6	21
Member	Connection	GetCategory() : ArrayList	55	3		7	18
Member	Connection	GetEbyServal(s tring) : int	59	3		5	14
Member	Connection	GetEventDetail (int) : ArrayList	50	3		8	25
Member	Connection	GetFromVenue Info(int, int) : int	58	3		5	15
Member	Connection	GetIDbyUserna meandEmail(st ring, string) : int	59	3		5	14
Member	Connection	GetNumofUser ByUsernameA ndEmail(string,	60	3		5	13

		string) : int					
Member	Connection	GetNumofUser ByUsernameA ndPassword(st ring, string) : int	59	3		5	13
Member	Connection	GetOrderConfi rmation(int) : PurchaseOrder	56	3		6	17
Member	Connection	GetPaymentTy pe() : DataTable	61	2		6	12
Member	Connection	GetSubCat() : DataTable	61	2		6	12
Member	Connection	GetSubCategor yBycat(string) : DataTable	61	2		6	12
Member	Connection	GetUpcomingE vent() : ArrayList	49	3		9	27
Member	Connection	GetUserbyUser ID(string) : ArrayList	48	3		7	28
Member	Connection	GetUserbyUser nameandPass word(string, string) : ArrayList	48	3		7	28



Member	Connection	GetUserID(string) : int	59	3		5	14
Member	Connection	NumofCat(string, string) : int	59	3		5	14
Member	Connection	NumofPtype(string) : int	59	3		5	14
Member	Connection	pass(string, string) : string	58	3		6	15
Member	Connection	SuggestEvent() : DataTable	61	2		6	12
Member	Connection	Upcome_Event() : DataTable	61	2		6	12
Member	Connection	UpcomeByCat(string) : DataTable	61	2		6	12
Member	Connection	UpcomeBySerVal(string) : DataTable	61	2		6	12
Member	Connection	UpcomeBySubCat(string) : DataTable	61	2		6	12
Member	Connection	UpdateCustomerInfo(string, string, string, string, string, string, string, string, string,	63	2		5	9

		string) : void					
Member	Connection	UpdatePassByI D(string, string) : void	65	2		5	9
Member	Connection	UpdatePassby UnameEmail(st ring, string, string) : void	64	2		5	9
Member	Connection	UserRegister(st ring, string) : void	64	2		5	9
Type	Event		91	34	1	2	42
Member	Event	Category.get() : string	98	1		0	1
Member	Event	Category.set(st ring) : void	95	1		0	1
Member	Event	Date.get() : string	98	1		0	1
Member	Event	Date.set(string ) : void	95	1		0	1
Member	Event	Datetime.get() : DateTime	98	1		1	1
Member	Event	Datetime.set(D ateTime) : void	95	1		1	1

Member	Event	Desc.get() : string	98	1		0	1
Member	Event	Desc.set(string) : void	95	1		0	1
Member	Event	E_SubCat.get() : IQueryable<Event>	98	1		1	1
Member	Event	E_SubCat.set(IQueryable<Event>) : void	95	1		1	1
Member	Event	Event()	100	1		0	1
Member	Event	Event(string, string, string, string, string, string, string)	66	1		0	9
Member	Event	ID.get() : int	98	1		0	1
Member	Event	ID.set(int) : void	95	1		0	1
Member	Event	Minprice.get() : double	98	1		0	1
Member	Event	Minprice.set(double) : void	95	1		0	1
Member	Event	Name.get() :	98	1		0	1

		string					
Member	Event	Name.set(string) : void	95	1		0	1
Member	Event	Picture.get() : string	98	1		0	1
Member	Event	Picture.set(string) : void	95	1		0	1
Member	Event	SeatingChart.get() : string	98	1		0	1
Member	Event	SeatingChart.set(string) : void	95	1		0	1
Member	Event	Status.get() : string	98	1		0	1
Member	Event	Status.set(string) : void	95	1		0	1
Member	Event	Subcategory.get() : string	98	1		0	1
Member	Event	Subcategory.set(string) : void	95	1		0	1
Member	Event	Time.get() : string	98	1		0	1
Member	Event	Time.set(string) : void	95	1		0	1

Member	Event	Totalavailable.get() : int	98	1		0	1
Member	Event	Totalavailable.set(int) : void	95	1		0	1
Member	Event	Totalsold.get() : int	98	1		0	1
Member	Event	Totalsold.set(int) : void	95	1		0	1
Member	Event	Venue.get() : string	98	1		0	1
Member	Event	Venue.set(string) : void	95	1		0	1
Type	PaymentType		93	6	1	0	8
Member	PaymentType	ID.get() : int	98	1		0	1
Member	PaymentType	ID.set(int) : void	95	1		0	1
Member	PaymentType	Name.get() : string	98	1		0	1
Member	PaymentType	Name.set(string) : void	95	1		0	1
Member	PaymentType	PaymentType()	100	1		0	1

Member	PaymentType	PaymentType(int, string)	82	1		0	3
Type	Seat		91	15	1	0	22
Member	Seat	Event.get() : string	98	1		0	1
Member	Seat	Event.set(string) : void	95	1		0	1
Member	Seat	Eventid.get() : int	98	1		0	1
Member	Seat	Eventid.set(int) : void	95	1		0	1
Member	Seat	Id.get() : int	98	1		0	1
Member	Seat	Id.set(int) : void	95	1		0	1
Member	Seat	Price.get() : float	98	1		0	1
Member	Seat	Price.set(float) : void	95	1		0	1
Member	Seat	SCat.get() : string	98	1		0	1
Member	Seat	SCat.set(string) : void	95	1		0	1

Member	Seat	Seat(int, int, string, string, float, string, int)	68	1		0	8
Member	Seat	Snum.get() : int	98	1		0	1
Member	Seat	Snum.set(int) : void	95	1		0	1
Member	Seat	Status.get() : string	98	1		0	1
Member	Seat	Status.set(string) : void	95	1		0	1
Type	SeatCategory		94	9	1	0	9
Member	SeatCategory	level.get() : int	98	1		0	1
Member	SeatCategory	level.set(int) : void	95	1		0	1
Member	SeatCategory	minprice.get() : int	98	1		0	1
Member	SeatCategory	minprice.set(int) : void	95	1		0	1
Member	SeatCategory	SeatCategory()	100	1		0	1

Member	SeatCategory	totalavailable.get() : int	98	1		0	1
Member	SeatCategory	totalavailable.set(int) : void	95	1		0	1
Member	SeatCategory	totalsold.get() : int	98	1		0	1
Member	SeatCategory	totalsold.set(int) : void	95	1		0	1
Type	ShoppingCart		92	5	1	1	8
Member	ShoppingCart	AddSeatID(int) : void	95	1		1	1
Member	ShoppingCart	CustomerID.get() : int	98	1		0	1
Member	ShoppingCart	CustomerID.set(int) : void	95	1		0	1
Member	ShoppingCart	GetSeatID(int) : int	86	1		1	2
Member	ShoppingCart	ShoppingCart(int)	80	1		1	3
Type	Users		90	28	1	0	47
Member	Users	Address1.get() : string	98	1		0	1



Member	Users	Address1.set(s tring) : void	95	1		0	1
Member	Users	Address2.get() : string	98	1		0	1
Member	Users	Address2.set(s tring) : void	95	1		0	1
Member	Users	City.get() : string	98	1		0	1
Member	Users	City.set(string) : void	95	1		0	1
Member	Users	Email.get() : string	98	1		0	1
Member	Users	Email.set(strin g) : void	95	1		0	1
Member	Users	FirstName.get( ) : string	98	1		0	1
Member	Users	FirstName.set( string) : void	95	1		0	1
Member	Users	ID.get() : int	98	1		0	1
Member	Users	ID.set(int) : void	95	1		0	1
Member	Users	LastName.get() : string	98	1		0	1

Member	Users	LastName.set(string) : void	95	1		0	1
Member	Users	Password.get() : string	98	1		0	1
Member	Users	Password.set(string) : void	95	1		0	1
Member	Users	Phone.get() : string	98	1		0	1
Member	Users	Phone.set(string) : void	95	1		0	1
Member	Users	State.get() : string	98	1		0	1
Member	Users	State.set(string) : void	95	1		0	1
Member	Users	Username.get() : string	98	1		0	1
Member	Users	Username.set(string) : void	95	1		0	1
Member	Users	Users(int, string, string, string, string, string, string, string, string, string)	60	1		0	14

Member	Users	Users(string, string, string, string, string)	70	1		0	7
Member	Users	UserType.get() : string	98	1		0	1
Member	Users	UserType.set(string) : void	95	1		0	1
Member	Users	Zipcode.get() : string	98	1		0	1
Member	Users	Zipcode.set(string) : void	95	1		0	1
Type	Venue		90	20	1	0	37
Member	Venue	Address1.get() : string	98	1		0	1
Member	Venue	Address1.set(string) : void	95	1		0	1
Member	Venue	Address2.get() : string	98	1		0	1
Member	Venue	Address2.set(string) : void	95	1		0	1
Member	Venue	City.get() : string	98	1		0	1
Member	Venue	City.set(string)	95	1		0	1

		: void					
Member	Venue	Desc.get() : string	98	1		0	1
Member	Venue	Desc.set(string) : void	95	1		0	1
Member	Venue	Id.get() : int	98	1		0	1
Member	Venue	Id.set(int) : void	95	1		0	1
Member	Venue	Name.get() : string	98	1		0	1
Member	Venue	Name.set(string) : void	95	1		0	1
Member	Venue	SeatingChart.get() : string	98	1		0	1
Member	Venue	SeatingChart.set(string) : void	95	1		0	1
Member	Venue	State.get() : string	98	1		0	1
Member	Venue	State.set(string) : void	95	1		0	1
Member	Venue	Venue(int, string, string, string, string, string, string)	65	1		0	10

		string, string)					
Member	Venue	Venue(string, string, string, string, string, string, string)	66	1		0	9
Member	Venue	Zip.get() : string	98	1		0	1
Member	Venue	Zip.set(string) : void	95	1		0	1
Namespace			72	40	4	47	102
Type	Default		71	6	4	13	13
Member	Default	Default()	100	1		1	1
Member	Default	GetImages() : void	59	3		7	10
Member	Default	Page_Load(object, EventArgs) : void	88	2		2	2
Type	E_detail		60	18	4	39	46
Member	E_detail	Buybtn_Click(object, EventArgs) : void	55	8		18	14

Member	E_detail	DisplayEventDetail() : void	57	2		13	12
Member	E_detail	E_detail()	92	1		2	1
Member	E_detail	Page_Load(object, EventArgs) : void	59	5		12	11
Member	E_detail	TicketSer_Click(object, EventArgs) : void	63	2		17	8
Type	E_DetailErr		100	2	4	2	1
Member	E_DetailErr	E_DetailErr()	100	1		1	1
Member	E_DetailErr	Page_Load(object, EventArgs) : void	100	1		1	0
Type	E_SubCat		56	14	4	13	42
Member	E_SubCat	D_Binding() : void	42	11		10	38
Member	E_SubCat	E_SubCat()	100	1		1	1
Member	E_SubCat	Page_Load(object, EventArgs) : void	78	2		4	3
Namespace			94	7	1	0	7

Type	EventSeat		94	7	1	0	7
Member	EventSeat	Capacity.get() : int	98	1		0	1
Member	EventSeat	Capacity.set(int) : void	95	1		0	1
Member	EventSeat	evenid.get() : int	98	1		0	1
Member	EventSeat	evenid.set(int) : void	95	1		0	1
Member	EventSeat	EventSeat()	100	1		0	1
Member	EventSeat	levelID.get() : int	98	1		0	1
Member	EventSeat	levelID.set(int) : void	95	1		0	1
Namespace			87	177	2	54	351
Type	CartItem		93	13	1	3	13
Member	CartItem	CartId.get() : string	98	1		0	1
Member	CartItem	CartId.set(string) : void	95	1		0	1
Member	CartItem	CartItem()	100	1		0	1

Member	CartItem	DateCreated.get() : DateTime	98	1		1	1
Member	CartItem	DateCreated.set(DateTime) : void	95	1		1	1
Member	CartItem	ItemId.get() : string	98	1		0	1
Member	CartItem	ItemId.set(string) : void	95	1		0	1
Member	CartItem	Product.get() : Product	98	1		1	1
Member	CartItem	Product.set(Product) : void	95	1		1	1
Member	CartItem	ProductId.get() : int	98	1		0	1
Member	CartItem	ProductId.set(int) : void	95	1		0	1
Member	CartItem	Quantity.get() : int	98	1		0	1
Member	CartItem	Quantity.set(int) : void	95	1		0	1
Type	Category		94	9	1	6	9
Member	Category	Category()	100	1		0	1



Member	Category	CategoryID.get() (): int	98	1		0	1
Member	Category	CategoryID.set(int) (): void	95	1		0	1
Member	Category	CategoryName.get() (): string	98	1		0	1
Member	Category	CategoryName.set(string) (): void	95	1		0	1
Member	Category	Description.get() (): string	98	1		0	1
Member	Category	Description.set(string) (): void	95	1		0	1
Member	Category	Products.get() : ICollection<Product>	98	1		2	1
Member	Category	Products.set(ICollection<Product>) (): void	95	1		2	1
Type	Order		92	33	1	10	33
Member	Order	Address.get() (): string	98	1		0	1
Member	Order	Address.set(string) (): void	95	1		0	1

Member	Order	City.get() : string	98	1		0	1
Member	Order	City.set(string) : void	95	1		0	1
Member	Order	Country.get() : string	98	1		0	1
Member	Order	Country.set(str ing) : void	95	1		0	1
Member	Order	Email.get() : string	98	1		0	1
Member	Order	Email.set(strin g) : void	95	1		0	1
Member	Order	FirstName.get( ) : string	98	1		0	1
Member	Order	FirstName.set( string) : void	95	1		0	1
Member	Order	HasBeenShipp ed.get() : bool	98	1		0	1
Member	Order	HasBeenShipp ed.set(bool) : void	95	1		0	1
Member	Order	LastName.get() : string	98	1		0	1
Member	Order	LastName.set(s	95	1		0	1

		tring) : void					
Member	Order	Order()	100	1		0	1
Member	Order	OrderDate.get() ): DateTime	98	1		1	1
Member	Order	OrderDate.set( DateTime) : void	95	1		1	1
Member	Order	OrderDetails.g et() : List<OrderDeta il>	98	1		2	1
Member	Order	OrderDetails.s et(List<OrderD etail>) : void	95	1		2	1
Member	Order	OrderId.get() : int	98	1		0	1
Member	Order	OrderId.set(int ) : void	95	1		0	1
Member	Order	PaymentTrans actionId.get() : string	98	1		0	1
Member	Order	PaymentTrans actionId.set(str ing) : void	95	1		0	1
Member	Order	Phone.get() : string	98	1		0	1

Member	Order	Phone.set(string) : void	95	1		0	1
Member	Order	PostalCode.get() : string	98	1		0	1
Member	Order	PostalCode.set(string) : void	95	1		0	1
Member	Order	State.get() : string	98	1		0	1
Member	Order	State.set(string) : void	95	1		0	1
Member	Order	Total.get() : decimal	98	1		1	1
Member	Order	Total.set(decimal) : void	95	1		1	1
Member	Order	Username.get() : string	98	1		0	1
Member	Order	Username.set(string) : void	95	1		0	1
Type	OrderDetail		93	13	1	1	13
Member	OrderDetail	OrderDetail()	100	1		0	1
Member	OrderDetail	OrderDetailId.get() : int	98	1		0	1

Member	OrderDetail	OrderDetailId.set(int) : void	95	1		0	1
Member	OrderDetail	OrderId.get() : int	98	1		0	1
Member	OrderDetail	OrderId.set(int) : void	95	1		0	1
Member	OrderDetail	ProductId.get() : int	98	1		0	1
Member	OrderDetail	ProductId.set(int) : void	95	1		0	1
Member	OrderDetail	Quantity.get() : int	98	1		0	1
Member	OrderDetail	Quantity.set(int) : void	95	1		0	1
Member	OrderDetail	UnitPrice.get() : double?	98	1		1	1
Member	OrderDetail	UnitPrice.set(double?) : void	95	1		1	1
Member	OrderDetail	Username.get() : string	98	1		0	1
Member	OrderDetail	Username.set(string) : void	95	1		0	1
Type	Product		93	15	1	6	15

Member	Product	Category.get() : Category	98	1		1	1
Member	Product	Category.set(C ategory) : void	95	1		1	1
Member	Product	CategoryID.get ( ) : int?	98	1		1	1
Member	Product	CategoryID.set (int?) : void	95	1		1	1
Member	Product	Description.get ( ) : string	98	1		0	1
Member	Product	Description.set (string) : void	95	1		0	1
Member	Product	ImagePath.get ( ) : string	98	1		0	1
Member	Product	ImagePath.set (string) : void	95	1		0	1
Member	Product	Product()	100	1		0	1
Member	Product	ProductID.get ( ) : int	98	1		0	1
Member	Product	ProductID.set(i nt) : void	95	1		0	1
Member	Product	ProductName. get() : string	98	1		0	1

Member	Product	ProductName. set(string) : void	95	1		0	1
Member	Product	UnitPrice.get() : double?	98	1		1	1
Member	Product	UnitPrice.set(d ouble?) : void	95	1		1	1
Type	ProductCo ntext		93	11	2	7	11
Member	ProductCo ntext	Categories.get( ) : DbSet<Categor y>	98	1		2	1
Member	ProductCo ntext	Categories.set( DbSet<Categor y>) : void	95	1		2	1
Member	ProductCo ntext	OrderDetails.g et() : DbSet<OrderD etail>	98	1		2	1
Member	ProductCo ntext	OrderDetails.s et(DbSet<Orde rDetail>) : void	95	1		2	1
Member	ProductCo ntext	Orders.get() : DbSet<Order>	98	1		2	1
Member	ProductCo ntext	Orders.set(DbS et<Order>) :	95	1		2	1

		void					
Member	ProductContext	ProductContext()	98	1		1	1
Member	ProductContext	Products.get() : DbSet<Product>	98	1		2	1
Member	ProductContext	Products.set(DbSet<Product>) : void	95	1		2	1
Member	ProductContext	ShoppingCartItems.get() : DbSet<CartItem>	98	1		2	1
Member	ProductContext	ShoppingCartItems.set(DbSet<CartItem>) : void	95	1		2	1
Type	ProductDatabaseInitializer		71	6	2	9	12
Member	ProductDatabaseInitializer	GetCategories() : List<Category>	73	1		2	3
Member	ProductDatabaseInitializer	GetProducts() : List<Product>	68	1		3	3
Member	ProductDatabaseInitializer	ProductDatabase	100	1		2	1



	izer	seInitializer()					
Member	ProductDataBaseInitializer	Seed(ProductContext) : void	70	3		7	5
Type	ShoppingCartAction		62	37	1	34	115
Member	ShoppingCartAction	AddToCart(int) : void	47	2		16	29
Member	ShoppingCartAction	Dispose() : void	80	2		2	3
Member	ShoppingCartAction	EmptyCart() : void	67	3		16	5
Member	ShoppingCartAction	GetCart(HttpContext) : ShoppingCartAction	75	2		1	4
Member	ShoppingCartAction	GetCartId() : string	63	3		5	8
Member	ShoppingCartAction	GetCartItems() : List<CartItem>	73	1		11	3
Member	ShoppingCartAction	GetCount() : int	69	2		10	4
Member	ShoppingCartAction	GetTotal() : decimal	65	3		12	5

Member	ShoppingCartAction	RemoveItem(string, int) : void	52	4		15	19
Member	ShoppingCartAction	ShoppingCartAction()	92	1		1	1
Member	ShoppingCartAction	ShoppingCartId.get() : string	98	1		0	1
Member	ShoppingCartAction	ShoppingCartId.set(string) : void	95	1		0	1
Member	ShoppingCartAction	UpdateItem(string, int, int) : void	52	4		14	19
Member	ShoppingCartAction	UpdateShoppingCartDatabase(string, ShoppingCartAction.ShoppingCartUpdates[]) : void	56	8		10	13
Type	ShoppingCartAction.ShoppingCartUpdates		100	0	1	0	0
Type	ShoppingCartActions		62	40	1	35	130
Member	ShoppingCartActions	AddToCart(int) : void	47	2		16	29

Member	ShoppingCartActions	Dispose() : void	81	2		2	3
Member	ShoppingCartActions	EmptyCart() : void	67	3		16	5
Member	ShoppingCartActions	GetCart(HttpContext) : ShoppingCartActions	75	2		1	4
Member	ShoppingCartActions	GetCartId() : string	63	3		5	8
Member	ShoppingCartActions	GetCartItems() : List<CartItem>	73	1		11	3
Member	ShoppingCartActions	GetCount() : int	69	2		10	4
Member	ShoppingCartActions	GetTotal() : decimal	65	3		12	5
Member	ShoppingCartActions	MigrateCart(string, string) : void	56	3		19	15
Member	ShoppingCartActions	RemoveItem(string, int) : void	52	4		15	19
Member	ShoppingCartActions	ShoppingCartActions()	92	1		1	1

Member	ShoppingCartActions	ShoppingCartId.get() : string	98	1		0	1
Member	ShoppingCartActions	ShoppingCartId.set(string) : void	95	1		0	1
Member	ShoppingCartActions	UpdateItem(string, int, int) : void	52	4		14	19
Member	ShoppingCartActions	UpdateShoppingCartDatabase(string, ShoppingCartActions.ShoppingCartUpdates[]) : void	56	8		10	13
Type	ShoppingCartActions. ShoppingCartUpdates		100	0	1	0	0
Namespace			78	46	4	47	173
Type	CheckoutCancel		96	3	4	4	2
Member	CheckoutCancel	Button1_Click(object, EventArgs) : void	94	1		3	1
Member	CheckoutCancel	CheckoutCancel()	100	1		1	1

Member	CheckoutCancel	Page_Load(object, EventArgs) : void	100	1		1	0
Type	CheckoutConfirmation		81	5	4	9	11
Member	CheckoutConfirmation	Button1_Click(object, EventArgs) : void	83	1		4	2
Member	CheckoutConfirmation	CheckoutConfirmation()	100	1		1	1
Member	CheckoutConfirmation	ConfirmationOrder() : void	66	2		7	7
Member	CheckoutConfirmation	Page_Load(object, EventArgs) : void	100	1		1	1
Type	CheckoutError		90	4	4	7	4
Member	CheckoutError	Button1_Click(object, EventArgs) : void	94	1		3	1
Member	CheckoutError	CheckoutError()	100	1		1	1

Member	CheckoutError	Page_Load(object, EventArgs) : void	81	2		5	2
Type	CheckoutReview		47	34	4	44	156
Member	CheckoutReview	CheckoutReview()	100	1		1	1
Member	CheckoutReview	DisplayTicketDetail() : void	36	7		20	63
Member	CheckoutReview	MakePayment_Click(object, EventArgs) : void	43	9		17	36
Member	CheckoutReview	Page_Load(object, EventArgs) : void	71	3		5	5
Member	CheckoutReview	TicketList_ItemCommand(object, DataListCommandEventArgs) : void	42	10		24	37
Member	CheckoutReview	timer1_tick(object, EventArgs) : void	55	4		12	14
Namespace			94	9	1	2	9

Type	PurchaseOrder		94	9	1	2	9
Member	PurchaseOrder	confirmationCode.get() : string	98	1		0	1
Member	PurchaseOrder	confirmationCode.set(string) : void	95	1		0	1
Member	PurchaseOrder	orderid.get() : int	98	1		0	1
Member	PurchaseOrder	orderid.set(int) : void	95	1		0	1
Member	PurchaseOrder	ordertotal.get() : decimal	98	1		1	1
Member	PurchaseOrder	ordertotal.set(decimal) : void	95	1		1	1
Member	PurchaseOrder	purchasedate.get() : DateTime	98	1		1	1
Member	PurchaseOrder	purchasedate.set(DateTime) : void	95	1		1	1
Member	PurchaseOrder	PurchaseOrder()	100	1		0	1
Namesp			59	61	4	30	194

ace							
Type	FogotPass		66	7	4	13	28
Member	FogotPass	FGPaSubmit_Click(object, EventArgs) : void	51	3		9	22
Member	FogotPass	FogotPass()	100	1		1	1
Member	FogotPass	IsValidEmail(string) : bool	75	2		1	5
Member	FogotPass	Page_Load(object, EventArgs) : void	100	1		1	0
Type	Login		61	13	4	18	49
Member	Login	ForgotPass_Click(object, EventArgs) : void	94	1		3	1
Member	Login	Login()	100	1		1	1
Member	Login	LoginButton_Click(object, EventArgs) : void	40	9		14	46
Member	Login	Page_Load(object, EventArgs) : void	100	1		1	0



Member	Login	RequestRegister_Click(object, EventArgs) : void	94	1		3	1
Type	Register		54	23	4	22	64
Member	Register	IsValidEmail(string) : bool	75	2		1	5
Member	Register	Page_Load(object, EventArgs) : void	100	1		1	0
Member	Register	Register()	100	1		1	1
Member	Register	RegisterButton_Click(object, EventArgs) : void	36	19		17	58
Type	ResetPassword		56	18	4	21	53
Member	ResetPassword	Page_Load(object, EventArgs) : void	53	5		8	16
Member	ResetPassword	RePaCancel_Click(object, EventArgs) : void	78	1		4	3
Member	ResetPassword	RePaSubmit_Click(object, EventArgs) :	44	11		15	33

		void					
Member	ResetPass word	ResetPassword ( )	100	1		1	1

## 8.2. SOFTWARE DEVELOPMENT LIFE CYCLE METHOD-LEVEL METRICS

These are all the method-level metrics for the SDLC solution created from the main spreadsheet of all the SDLC metrics.

**Table 13: SDLC Method-Level Metrics**

Scope	Approach	Maintainability Index	Cyclomatic Complexity	Depth of Inherit	Class Coupling	Lines of Code
-------	----------	-----------------------	-----------------------	------------------	----------------	---------------

				ance		
Method	SDLC	57	27	4	15	111
Method	SDLC	76	14	3	5	32
Method	SDLC	75	2	1	5	6
Method	SDLC	95	2	1	2	2
Method	SDLC	84	2	2	8	5
Method	SDLC	90	2	1	4	3
Method	SDLC	62	26	5	28	65
Method	SDLC	100	2	4	2	1
Method	SDLC	58	32	4	25	122
Method	SDLC	100	2	4	2	1
Method	SDLC	73	15	5	14	41
Method	SDLC	69	9	4	22	28
Method	SDLC	61	46	4	61	147
Method	SDLC	71	6	4	13	16
Method	SDLC	57	42	4	54	152

Method	SDLC	92	9	1	0	12
Method	SDLC	84	5	1	1	12
Method	SDLC	93	17	1	3	17
Method	SDLC	92	7	1	0	10
Method	SDLC	57	109	1	32	613
Method	SDLC	91	34	1	2	42
Method	SDLC	93	6	1	0	8
Method	SDLC	91	15	1	0	22
Method	SDLC	94	9	1	0	9
Method	SDLC	92	5	1	1	8
Method	SDLC	90	28	1	0	47
Method	SDLC	90	20	1	0	37
Method	SDLC	71	6	4	13	13
Method	SDLC	60	18	4	39	46
Method	SDLC	100	2	4	2	1
Method	SDLC	56	14	4	13	42

Method	SDLC	94	7	1	0	7
Method	SDLC	93	13	1	3	13
Method	SDLC	94	9	1	6	9
Method	SDLC	92	33	1	10	33
Method	SDLC	93	13	1	1	13
Method	SDLC	93	15	1	6	15
Method	SDLC	93	11	2	7	11
Method	SDLC	71	6	2	9	12
Method	SDLC	62	37	1	34	115
Method	SDLC	100	0	1	0	0
Method	SDLC	62	40	1	35	130
Method	SDLC	100	0	1	0	0
Method	SDLC	96	3	4	4	2
Method	SDLC	81	5	4	9	11
Method	SDLC	90	4	4	7	4
Method	SDLC	47	34	4	44	156

Method	SDLC	94	9	1	2	9
Method	SDLC	66	7	4	13	28
Method	SDLC	61	13	4	18	49
Method	SDLC	54	23	4	22	64
Method	SDLC	56	18	4	21	53

### **8.3. SOFTWARE DEVELOPMENT LIFE CYCLE CLASS-LEVEL METRICS**

These are all the class-level metrics for the SDLC solution created from the main spreadsheet of all the SDLC metrics.

**Table 14: SDLC Class-Level Metrics**

Scope	Approach	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Class	SDLC	84	36	5	48	82
Class	SDLC	58	32	4	25	122
Class	SDLC	75	129	5	81	397
Class	SDLC	88	255	1	35	825

Class	SDLC	72	40	4	47	102
Class	SDLC	94	7	1	0	7
Class	SDLC	87	177	2	54	351
Class	SDLC	78	46	4	47	173
Class	SDLC	94	9	1	2	9
Class	SDLC	59	61	4	30	194

#### **8.4. SOFTWARE DEVELOPMENT LIFE CYCLE PROJECT-LEVEL METRICS**

These are all the metrics for the SDLC solution extracted from Visual Studio 2013 Code Analyzer.

**Table 15: SDLC Project-Level Metrics**

Scope	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Project	80	833	5	210	2405

## 9. APPENDIX .NET CODE ANALYSIS TOOL

This appendix presents only the tool used for automating metric collection and analysis.

### 9.1. .NET CODE ANALYSIS TOOL

The .NET Compiler Platform is a set of open-source compilers and code analysis APIs for C# and Visual Basic.NET languages from Microsoft. The project notably includes self-hosting versions of the C# and VB.NET compilers – compilers written in the languages themselves. The compilers are available via the customary command-line programs but also as APIs available locally from within .NET code. The tool exposes modules for analysis of code and also dynamic compilation. Correctness, performance and maintainability are all involved in creating quality code. Visual Studio diagnostic tools can help you to develop and maintain high standards of code.



## **10. APPENDIX D METRICS**

The following list shows the code metrics results that Visual Studio calculates:

### **10.1. MAINTAINABILITY INDEX**

The maintainability index value is between 0 and 100 and signifies the comparative way of sustaining the code. The higher the value the better the maintainability. A good maintainability value is usually between 20 and 100. Between 10 and 19 shows that the code is reasonably maintainable. Between 0 and 9 and indicates poor maintainability (Code Metrics Results, 2015).

### **10.2. CYCLOMATIC COMPLEXITY**

Measures the structural complexity of the code. The flow of the program, how it breaks into different directions is a measure of the cyclomatic complexity. This measurement involves the way the program loops and branches. The more the complexity the more the test coverage to completely test the code and also the more difficulties maintaining the code (Code Metrics Results, 2015).

### **10.3. DEPTH OF INHERITANCE**

Depth of inheritance designates the number of class definitions that spread to the root of the class structure. If the root is deeper, the structure will be more difficult to understand the methods involved (Code Metrics Results, 2015).

#### **10.4. CLASS COUPLING**

Measures how many classes directly depend on a unique class. This is actually measuring the link between objects in a class. It is more desirable to have high cohesion and low coupling. High coupling will be difficult to reuse and maintain because of its many linkages to other classes or objects (Code Metrics Results, 2015).

#### **10.5. LINES OF CODE**

This is an approximation of number of lines in the code. The count is based on the intermediate language (IL) compiled during metric generation, so it is not the exact count of the number of lines in a source code. When the count is very high, there is a problem of maintainability due to many line to work with and difficulties understanding the code (Code Metrics Results, 2015).