



Agile Software Development of Embedded Systems

Version : 1.0
Date : 2006.03.09
Pages : 15

Authors

Maria Sinaalto

Status

Final

Confidentiality

Public

Agile Deliverable D.5.2.10

The impact of test- driven development on design quality

Abstract

This document contains the results of a multiple case study on the impact of test-driven development on design quality. The empirical data presented in the latter part of this document is based on the case projects conducted at VTT Technical Research Centre of Finland in October-December 2005.



I T E A

INFORMATION TECHNOLOGY
FOR EUROPEAN ADVANCEMENT

	<p style="text-align: center;">Test-Driven Development Deliverable ID:</p>	Page :2 of 15
		1.0 Date : 09.03.06
		Status : Final Confid : Public

TABLE OF CONTENTS

1.	INTRODUCTION	4
2.	TEST-DRIVEN DEVELOPMENT	5
3.	EMPIRICAL RESEARCH DESIGN.....	6
3.1	RESEARCH SETTING.....	6
3.2	DATA COLLECTION AND METRICS.....	6
4.	CASE STUDY RESULTS	8
4.1	CASE PROJECT OVERVIEW	8
4.2	THE RESULTS OF THE QUALITY METRICS.....	8
4.3	INTERVIEW RESULTS.....	10
4.3.1	<i>Adoption of TDD.....</i>	<i>10</i>
4.3.2	<i>Advantages /disadvantages of TDD</i>	<i>10</i>
4.3.3	<i>Other upcoming issues related to TDD</i>	<i>10</i>
5.	CONCLUSIONS.....	11
6.	REFERENCES	12



Test-Driven Development
Deliverable ID:

Page :3 of 15

1.0
Date : 09.03.06


Status : Final
Confid : Public

CHANGE LOG

Ver s.	Date	Author	Description
0.1	3.1.06	Maria Siniaalto	First draft created
0.2	13.1.06	Maria Siniaalto	Case study results and appendices added
0.3	17.1.06	Maria Siniaalto	First full draft
1.0	9.3.06	Maria Siniaalto	Reviewed by Tuomas Ihme

APPLICABLE DOCUMENT LIST

Ref.	Title, author, source, date, status	Identification

	<p style="text-align: center;">Test-Driven Development Deliverable ID:</p>	Page :4 of 15
		1.0 Date : 09.03.06
		Status : Final Confid : Public

1. INTRODUCTION

This document contains the results of the multiple case study on the impact of Test-driven development (TDD) on design quality. The empirical content of this document is based on the evidence obtained from three case projects conducted within VTT Technical Research Centre of Finland.

This document is structured as follows: First, Test-driven development technique is introduced followed by a chapter presenting the design of the research along with the used metrics. Next, the results are presented and, finally, conclusions are drawn.



2. TEST-DRIVEN DEVELOPMENT

Test-driven development (TDD) is a core part of the Agile code development approach derived from Extreme Programming (XP) (Beck 2004) and the principles of the Agile Manifesto (Beck, Beedle, et al.). An early reference to the use of TDD is the NASA Project Mercury in the 1960's (Larman and Basili 2003), so the practice is not all that new.

Despite its name, TDD is rather a development and design technique than a testing technique. TDD is an incremental development technique in which the tests are written prior to the production code (Beck 2001). When the test is passed, the code is refactored to improve the internal structure of the code. This incremental cycle presented in Figure 1 is repeated until all functionality is implemented (Astels 2003).

In literature, TDD is proposed to guarantee testability and to reach exhaustive test coverage (Astels 2003). It is also claimed to increase developer confidence (Beck 2003) and to enable integration more often that allows larger teams of programmers to work on the same code base, because the code can be checked in more often. TDD is said to affect the design of the code by helping to produce highly cohesive and loosely coupled systems. It is also said to encourage the explicitness about the scope of the implementation while it helps separating the logical and physical design, when only the code needed at a certain time is implemented. (Beck 2001)

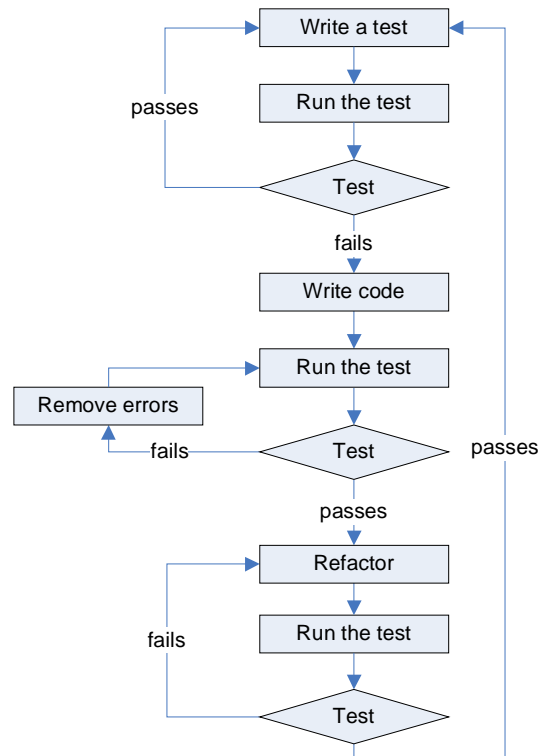



Figure 1. TDD cycle.

	Test-Driven Development Deliverable ID:	Page :6 of 15
		1.0 Date : 09.03.06
		Status : Final Confid : Public

3. EMPIRICAL RESEARCH DESIGN

3.1 RESEARCH SETTING

A team of six developers implemented an Agile project management tool to be used via web browser in a project called Avalanche. Five of the developers participated in the implementation fulltime while the author of this paper provided technical support and acted as a project manager. All developers were fifth or sixth year university students and one of them had industrial software development experience. The team worked in co-located development environment and used an agile software development method for mobile application development, Mobile-D™ (Ihme and Abrahamsson 2005). This paper focuses on TDD, which was the emphasized practice of the used approach.

A support and steering group of nine people, including the customer's representatives and consultants of different areas, was also formed for the project. The supporting tasks of the TDD consisted of selecting of the tools that enable fast use of TDD, providing training for the use of the approach, monitoring the TDD process during the project and assisting in problem situations.

3.2 DATA COLLECTION AND METRICS

The design quality was assessed through six object-oriented (OO) metrics whereas the developer experiences were determined based on a final interview held after the Avalanche project. The metrics were gathered from two other case projects , eXpert and zOmbie, using iterative test-last (ITL) approach in order to make comparison between TDD and ITL.

In software development, design is the earliest stage on which the structure of the system is defined, and therefore, term "design" is generally used when referring to architectural design. The traditional product metrics such as size, complexity and performance are not sufficient for characterizing and assessing the quality of OO software systems, and they should be supplemented with notions such as encapsulation and inheritance, which are inherent in object-orientation (Basili and Melo 1996). The OO design metrics for measuring the design quality effects of TDD within this study were selected from the ones proposed by Chidamber and Kemerer (Chidamber and Kemerer 1994): weighted methods per class (WMC), depth of inheritance tree (DIT), number of children (NOC), coupling between objects (CBO) and response for a class (RFC). Lack of cohesion in methods (LCOM) was, however, replaced with the LCOM* proposed by Henderson-Sellers (Henderson-Sellers 1996), because the original LCOM does not always provide valid results since the equation is badly defined (Basili and Melo 1996). The used thresholds are the ones proposed by Rosenberg (Rosenberg, Stapko, et al. 1999).

Table 1 summarizes the used metrics. The WMC value presents class's complexity, and it can be used for predicting how much effort certain class requires when it is maintained. A high value of WMC implies that the class is probably less reusable and requires more maintaining effort, because it is likely to have a greater impact on its derived classes and it may be more application specific. The proposed threshold for WMC is 20 when the value presents the number of methods.

The DIT value measures the depth of each class within its hierarchy, and it can be utilized for predicting the complexity and the potential reuse of a class. A high value of DIT indicates increased complexity, because more methods are involved, but also increased potential for reuse of inherited methods. A proposed threshold for DIT is around 6.

NOC presents the number of class's immediate subclasses, and it may be used for assessing class's potential reuse and the effort the class requires when it is tested. A high value of NOC may be a sign of an improper use of subclassing and it implies that the class may require more testing, because it has many children using its methods. On the other hand, a high NOC value indicates greater



Test-Driven Development
Deliverable ID:

Page :7 of 15

1.0
Date : 09.03.06

Status : Final
Confid : Public

potential for reuse, because inheritance is one form of reuse. There is no specific threshold proposed for NOC, at least to the author's knowledge.

CBO presents the number of classes to which a particular class is coupled. A high CBO value indicates increased testing effort and decreased reuse possibilities, because excessive coupling between object classes decreases design modularity and makes the system more sensitive to the changes. A threshold value of 5 is proposed for the value of CBO.

The value of RFC presents the number of methods that can be invoked in response to a message to an object of the class or by some method in the class. The testing and debugging of a class with a high value of RFC is difficult, because the increased amount of communication between classes requires more understanding on the part of the tester. It also indicates increased complexity. A threshold of 50 is proposed for RFC, although the value is acceptable up to 100.

LCOM* measures the correlation between the methods and the local variables of a class. The value varies from 0 to 1, with the value zero representing perfect cohesion and with the value one presenting extreme lack of cohesion. A high LCOM* value indicates decreased encapsulation and increased complexity while the low value implies high cohesion and well design. Systems with low LCOM* are likely to be simple and promote reuse whereas systems with high LCOM* are probably more error-prone.

Table 1. Quality Metrics

Metric	Threshold	Description
WMC	20	A count of the methods of a class or the sum of the complexities of the methods. High value of WMC may indicate decreased reusability and maintainability
DIT	6	The depth of a class in an inheritance hierarchy. High value of DIT may indicate increased reusability, but also increased complexity.
NOC	-	The number of immediate subclasses subordinate to a class in the hierarchy. High value of NOC may indicate increased reusability, but also increased testing effort and it can be a sign of a misuse of subclassing.
CBO	5	Two classes are coupled to each other when methods declared in one class use methods or instance variables defined by the other class. A high value of CBO may indicate limited reuse possibilities and decreased modularity in the design.
RFC	50 (acceptable up to 100)	The combination of the complexity of a class through the number of methods and the amount of communication with other classes. High value of RFC may indicate increased testing effort and complexity.
LCOM*	[0,1]	The degree to which the methods within a class are related to one another. A low value of LCOM* may indicate a good design and increased reusability, where as a high value may indicate decreased encapsulation and increased complexity.



4. CASE STUDY RESULTS

4.1 CASE PROJECT OVERVIEW

Three case projects were included in this empirical evaluation. All projects were conducted within VTT Electronics Industry Driven Experimental Software Engineering Initiative (ENERGI), which is a framework for conducting software projects with both research and business objectives.

Table 2 provides an overview on the case projects included in the evaluation. The projects were selected based on the solution architecture and the programming language (Java). All projects continued approximately for nine weeks. Iterative test-last development (ITL) technique was used in the eXpert and zOmbie projects whereas Test-driven development was used in the Avalanche project. To make the comparison more even, only the "corresponding" parts (i.e. providing similar services) of the server side software from each project were included.

The subjects in all projects were 5th to 6th year university students. The eXpert project extended from February to March 2003. A web-based virtual file system for managing the project documents was the concrete product implemented in the project. The system was realized with Java and Java Server Pages (JSP) on a MySQL database.

The zOmbie project extended from October to November 2003, and a mobile application for browsing stock markets using PDA or mobile phone as the client terminal was the concrete product implemented in the project. The software was implemented using Java and Java Servlets on the server side and Mobile Information Device Profile (MIDP) 1.0 on the client side.


The Avalanche project extended from October to December 2005. An agile project management tool to be used via a web browser and mobile phone was the concrete product realized in the project. The software was implemented on a MySQL database using Java and Java Servlets on the server side and MIDP 2.0 on the client side. Graphical user interface (GUI) was implemented with Flash.

Table 2. Summary of the case projects

Item	eXpert	zOmbie	Avalanche
Team size	4	5	6
Iterations (weeks)	3x2 3x1	1x1 3x2 2x1	1x1 2x2 1x3 1x1
Development Technique	ITL	ITL	TDD
Total LOC	7700	7000	5800 (does not contain GUI)
LOC included in the comparison	2575	1971	971

4.2 THE RESULTS OF THE QUALITY METRICS

The results of the metrics are presented in APPENDIX A. The WMC values of all three case projects are below the proposed threshold: The eXpert project has the highest value of WMC while the zOmbie project has the lowest value of WMC. Although the WMC values do not indicate decreased reusability or maintainability in any of the case projects, there are indications that the

	Test-Driven Development Deliverable ID:	Page :9 of 15
		1.0 Date : 09.03.06
		Status : Final Confid : Public

zOmbie project might be slightly more reusable and require less maintaining effort than the eXpert or the Avalanche projects. This result is not likely to be due to TDD or ITL, but is rather due to chance or the individual developer experiences and skills.

The DIT values of all case projects are below the proposed threshold, and somewhat equal. The DIT values indicate neither better reusability nor increased complexity in any of the case projects. It can be only concluded that the inheritance was not really used in any of the projects when implementing the code.

The NOC values of the Avalanche project are many times higher than the NOC values of the eXpert or the zOmbie projects. Since the individual values for classes are very low in all the projects, the differences between Avalanche and the two ITL projects are more likely to be due a chance than TDD.


The CBO values of all case projects are below the proposed threshold. The values of the zOmbie project are the lowest that may imply that zOmbie has better reuse possibilities and requires less testing effort when compared to eXpert or Avalanche. However, this is more likely to be due to chance or the differences in the individual developer skills and experiences than due to TDD or ITL, because the CBO values of the eXpert project, which also used ITL, are the highest. Based on the CBO results, there were no indications that the reuse possibilities would be limited or the design modularity would be decreased in any of the projects.

The values of RFC are below the proposed threshold in all the case projects, which implies that the complexities and testing effort are not increased in any of the projects. The Avalanche project has the lowest value of RFC, but the differences when compared to the eXpert or the zOmbie projects are not significant. When compared to the eXpert project, the Avalanche project is probably less complex and requires less testing effort, but the same conclusions cannot be drawn when the RFC values of Avalanche are compared to the RFC values of zOmbie.

The values WMC, DIT, NOC, CBO and RFC of all the projects are below the proposed thresholds. In addition, the results are partly contradicting, i.e. the values of the zOmbie project are the highest while the values of the eXpert project are the lowest when examining some of the metric results, which implies that the result is not due to ITL or TDD. Based on these findings, it can be concluded that there are no significant differences between Test-driven development and iterative test-last approaches when comparing the quality of the design using the results of WMC, DIT, NOC, CBO or RFC. The minor differences revealed by these metrics are more likely to be due to chance, the differences in the individual developer skills or the characteristics of the systems implemented during projects.

The only metric providing some variance is the result of LCOM*, which is clearly contrary to the claims presented in literature: Avalanche has significantly higher LCOM* value indicating decreased encapsulation and increased complexity while TDD is proposed to produce highly cohesive systems (Beck 2003). The low cohesion was found distributed around the whole system in the Avalanche project, which indicates that it is not likely to be due to a chance. Based on this result, it can be concluded that TDD does not necessary help to produce highly cohesive code when junior developers employ it.

In addition to the above presented results, the code coverage was determined for the Avalanche project. The test coverage of the eXpert and zOmbie projects was not available. However, the test coverage in the Avalanche project (on the unit test level) was extremely high, 97.7 %.

	Test-Driven Development Deliverable ID:	Page :10 of 15
		1.0 Date : 09.03.06
		Status : Final Confid : Public

4.3 INTERVIEW RESULTS

The team members of the Avalanche project were interviewed one at a time after the project. One developer, who only implemented GUI and did not use TDD, was not included in the interview, and therefore the answers are obtained from a total of four developers. The interview was performed according to the themes presented in Table 3. Only themes instead of questions were assigned for the interview to allow the developers to express their experiences and feelings freely. In the following subsections, the results of the interviews are grouped together according to the themes of the interview.

Table 3. Themes of the final interview

Theme
1. Adoption of TDD
2. Advantages /disadvantages of TDD
3. Other upcoming issues related to TDD

4.3.1 Adoption of TDD


75 % of the developers found the adoption of the TDD difficult, and they estimated that it took approximately two weeks before TDD developed to a routine. They also thought that TDD requires a total change in the way of thinking and they would have needed more training on TDD or used development tools before the project started. However, some of the results related to the adoption of TDD were inconsistent: One developer told that the TDD required a total change in the way of thinking, but at the same time he thought that TDD is just a development technique, and does not differ so much from the others.

4.3.2 Advantages /disadvantages of TDD

The developers felt that TDD gave them courage to refactor and continue implementing despite possible problems, because they could always rely on the exhaustive test suite. In this context refactoring refers rather to the implementing of the new features or to the changing of the existing ones than to pure refactoring. 75 % of the developers told that they would like to continue using TDD, because it improves their work significantly. The developers also believed that TDD might decrease overall project lead-times, improve software quality and facilitate maintainability.

4.3.3 Other upcoming issues related to TDD


According to the interview findings, the tools that were used in the project were especially suitable for TDD and facilitated the developers work. One developer told that she would like to see TDD to be taught in university, because in her opinion it would probably facilitate and improve the students' work. Most of the developers reported also to be willing to continue to use TDD in the future. The only negative findings were related to the adoption of TDD.

	Test-Driven Development Deliverable ID:	Page :11 of 15
		1.0 Date : 09.03.06
		Status : Final Confid : Public

5. CONCLUSIONS

Based on the empirical findings, TDD does not seem to produce highly cohesive code as suggested in literature. All projects seemed to have produced loosely coupled code, regardless of what development practice was used, but the cohesion was significantly decreased in the project using TDD. However, it must be kept in mind that the design quality was only explored against one set of design metrics and the sample size on which the metrics were run was quite small. Another issue possibly affecting the results is the fact that the development team in the Avalanche project was much less experienced than the teams in eXpert and zOmbie: Only one of the developers had some industrial experience, whereas almost all developers in the eXpert and zOmbie projects had interest in coding or industrial coding experience. The experience of other developers in the Avalanche project was limited to the university's programming courses.

The design quality metrics employed in this work require mastery of the object-oriented development, and the skill level of the development team is certainly significant and affects the composition of the code. Another issue noticed during the project was that the refactoring should have been emphasized more and that the directions should have been more systematic. The qualitative findings based on the interview were as much a like as suggested in the literature: The developers found the adoption of TDD difficult and would have needed more training before the start of the project. They also felt that TDD gave them courage to refactor and confidence in their code, because they could proceed with the knowledge that the changes they make do not introduce new faults somewhere else in the system. Most of the developers were willing to continue to use TDD.

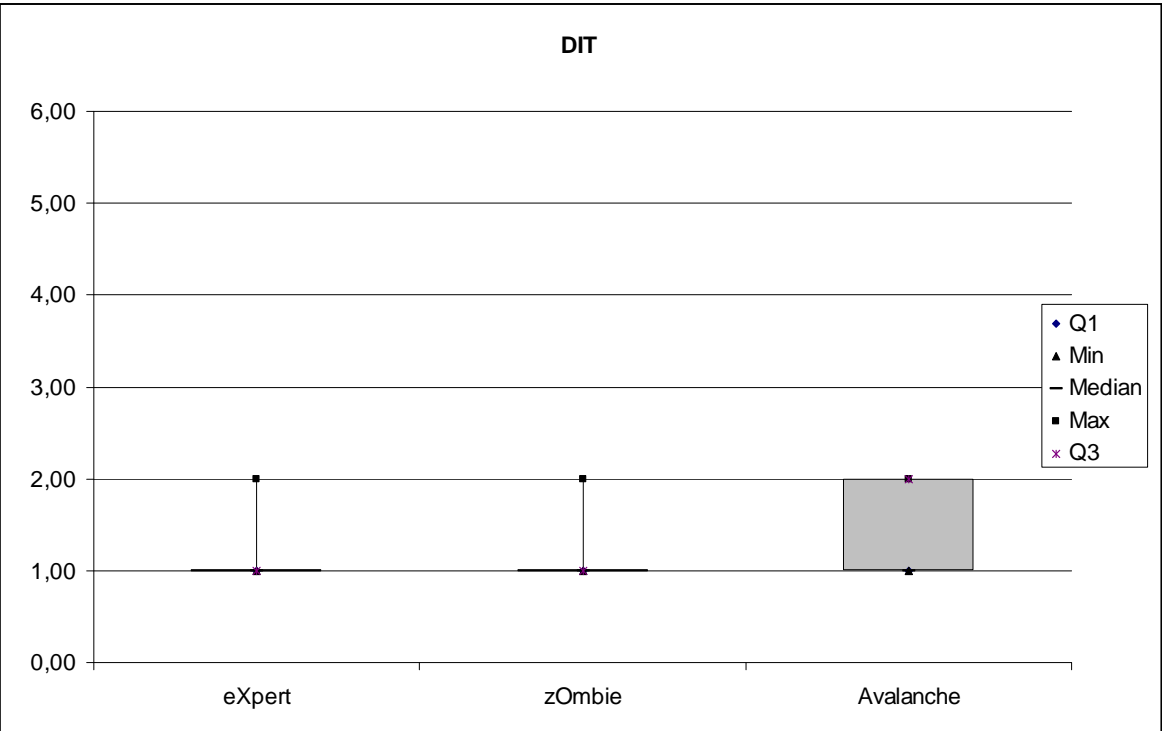
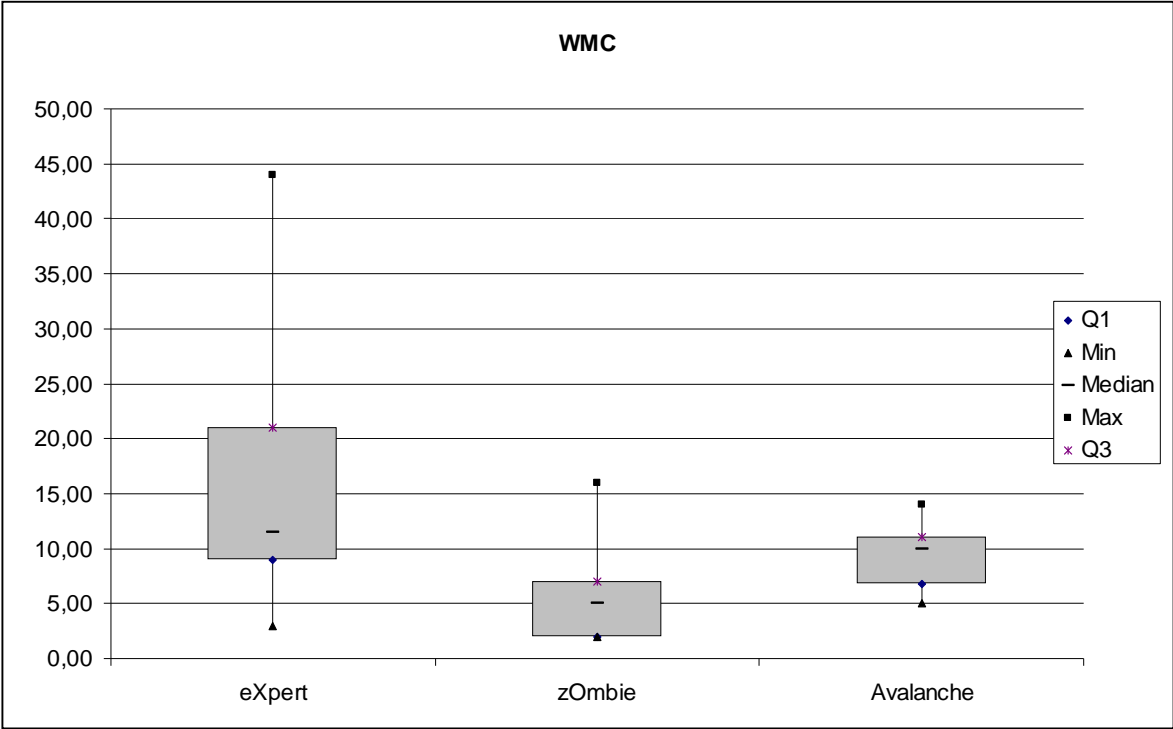
	Test-Driven Development Deliverable ID:	Page :12 of 15
		1.0 Date : 09.03.06
		Status : Final Confid : Public

6. REFERENCES

- Astels, D. (2003). Test-Driven Development: A Practical Guide. Upper Saddle River, New Jersey, USA, Prentice Hall.
- Basili, V. R. and W. L. Melo (1996). "A validation of Object-Oriented Design Metrics as Quality Indicators." IEEE Transactions on Software Engineering Proceedings **22**(10): 751 - 761.
- Beck, K. (2001). "Aim, fire." IEEE Software **18**(5): 87 - 89.
- Beck, K. (2003). Test-Driven Development By Example. Boston, Massachusetts, USA, Addison-Wesley.
- Beck, K. (2004). Extreme Programming Explained, Second Edition :Embrace Change. Boston, Massachusetts, USA, Addison-Wesley.
- Beck, K., M. Beedle, et al., Manifesto for Agile Software Development, 28.12.2005, <http://www.agilemanifesto.org>
- Chidamber, S. R. and C. F. Kemerer (1994). "A metrics Suite for Object Oriented Design." IEEE Transactions on Software Engineering **20**(6): 476 - 493.
- Henderson-Sellers, B. (1996). Object-Oriented Metrics: Measures of Complexity. Upper Saddle River, New Jersey, USA, Prentice Hall.
- Ihme, T. and P. Abrahamsson (2005). "Agile Architecting: The Use of Architectural Patterns in Mobile Java Applications." International Journal of Agile Manufacturing **8**(2).
- Larman, G. and V. R. Basili (2003). "Iterative and Incremental Development: A Brief History." IEEE Computer **36**(6): 47 - 56.
- Rosenberg, L. H., R. Stapko, et al. (1999). Applying Object Oriented Metrics. The Sixth International Symposium on Software Metrics, Boca Raton, Florida, USA.



APPENDIX A: THE RESULTS OF THE METRICS





Test-Driven Development
Deliverable ID:

