The Effect of Test-Driven Development on Program Code

Matthias M. Müller

Fakultät für Informatik, Universität Karlsruhe,
Am Fasanengarten 5, 76 131 Karlsruhe, Germany
muellerm@ipd.uka.de

Abstract. Usage of test-driven development (TDD) is said to lead to better testable programs. However, no study answers either the question how this better testability can be measured nor whether the feature of better testability exists. To answer both questions we present the concept of the controllability of assignments. We studied this metric on various TDD and conventional projects. Assignment controllability seems to support the rules of thumb for testable code, e.g. small classes with low coupling are better testable than large classes with high coupling. And as opposed to the Chidamber and Kemerer metric suite for object-oriented design, controllability of assignments seems to be an indicator whether a project was developed with TDD or not.

1 Introduction

Test-driven development (TDD) is besides pair programming one of the main programming techniques in extreme programming. However, test-driven development has not been studied as thoroughly as pair programming. Studies dealing with test-driven development have focused on the development cost or the quality of the written tests [1–5]. Nobody investigated the structure of programs developed with test-driven development although it is claimed that "Test-first code tends to be more cohesive and less coupled than code in which testing isn't part of the intimate coding cycle" [6, p. 88].

This paper uses the concept of controllability [7] to investigate the effect of test-driven development on program code. Controllability means that the program can be put in every legal state by only altering the inputs. This concept is applied to assignments. Controllability of an assignment means that the operands on the right hand side are input parameters of a method or these operands can be calculated from these parameters. We present a new metric called *assignment controllability* (AC) which quantifies this property for methods and classes. The assignment controllability is compared to the Chidamber and Kemerer metric suite for object-oriented design [8] using a set of TDD and open-source projects As a result, assignment controllability seems to support the rules of thumb of testable code, i.e. fewer number of methods and low coupling, and assignment controllability seems to be an indicator whether a project was developed using TDD or not. Throughout the paper we refer to projects which have been developed with test-driven development as TDD-projects.

2 The Metric

2.1 Controllability

Controllability is a concept from the design of digital circuits. For example Abramovici et al. [9] define controllability as 'the ability to establish a specific signal value at each node in a circuit by setting values on the circuit's inputs.' The transformation of controllability to an object oriented program means that all input parameters are known and that these parameters provide enough information to describe the state and the behaviour of the program. In this paper, we concentrate on assignments as they provide the only means to change the state of objects which represent the state of a program. Invocations of methods which do not return any value are ignored by our analysis, so far.

2.2 Controllability of Assignments

The calculation of controllability is a data-flow problem. First of all, all parameters of a method as well as private or public instance or class variables are controllable. These elements form the basic blocks for the calculation. Table 1 shows the rules for the remaining parts of an assignment. The result of an assignment, i. e. the left hand side, is controllable if its right hand side is controllable. An expression is controllable if all its identifiers are controllable. The conditional assignment is a special case, see Figure 1. The object a in line 6 is controllable only if either *both* expressions exp_1 and exp_2 in the lines 2 and 4 are controllable, or, the condition in line 1 and *one* of the expressions exp_1 or exp_2 is controllable. All constants and all messages send to `this` are not controllable.

Table 1. Controllability of Operations

Operation	**Controllability of the result**
$lhs := rhs$	The left hand side of an assignment is controllable if the right hand side is controllable.
$exp_1 \oplus exp_2$	The result of an arbitrary binary operation $\oplus$ is controllable, if both operands exp_1 and exp_2 are controllable.
$\oplus\ exp_1$	The result of an arbitrary unary operation $\oplus$ is controllable, if the operand exp_1 controllable.
$obj.foo(a, b)$	The result of a function call controllable, if obj and parameters a and b are controllable.

2.3 Calculation

The controllability of a method m is the ratio of controllable assignments to all assignments in m. We call this metric *Assignment Controllability AC*:

$$\mathsf{AC}(m) = \frac{\text{number of controllable assignments in method } m}{\text{number of all assignments in method } m}$$

```
1  if ( cond ) {
2    a = exp1;
3  } else {
4    a = exp2;
5  }
6  b = ... a ...
```

Fig. 1. Conditional Assignment

Its range varies between 0 and 1. The controllability of a class c is the average controllability of its methods. For a class c having n methods m_i $(i = 1 \dots n)$ the assignment controllability is

$$\mathsf{AC}(c) = \frac{1}{n} \sum_{i=1}^{i=n} \mathsf{AC}(m_i) \tag{1}$$

Methods without any assignments are ignored in the calculation.

A program to calculate the assignment controllability metric was implemented using the *Byte Code Engineering Library* (BCEL) [10] of the Jakarta Apache Project.

3 Data Set

Table 2 lists the projects used for this analysis. The type of project is given in

Table 2. Overview of Projects

Name	TDD	Number of Classes	Packages
Webtest	yes	149	21
XPChess1	yes	63	8
XPChess2	yes	48	8
XPChess3	yes	68	8
Yaps	yes	100	16
Sum		428	61
Ant	no	372	22
JUnit	no	75	7
Log4j	no	228	19
Sum		675	48

the second column. The columns 3 and 4 present the number of classes and the number of packages for each project. Webtest [11] is a testing tool for web applications. The projects XPChess1, XPChess2, and XPChess3 are student projects from the extreme programming lab course held in the summer term 2005 at the

Universität Karlsruhe. These programs are chess engines with command line interface. Yaps is a portal framework of a medium-sized company. Ant [12] is the Apache platform independent implementation of make. JUnit is the Java testing framework of the xUnit family. Log4j [13] is the Java implementation of the protocol framework from the Apache project. The number of classes and packages refer to the size of the application. The test classes were omitted because the test classes were not part of this study.

4 Results

4.1 Metrics used in this Study

The assignment controllability metric is compared to the following eight metrics. The first six metrics are known as the Chidamber and Kemerer metric suite for object oriented design [8]. The suite contains the weighted sum of methods of a class (WMC). As the weights of the sum are set to one, the weighted method per class metric simply presents the number of methods of a class. The depth of a class in the inheritance tree (DIT) is the next metric. The third metric is the number of children of a class (NOC). For the number of children only the direct subclasses are count. The coupling of a class c (CBO) is the number of classes from which c uses methods or variables. The response set of a class c (RFC) is the number of all methods which are called directly from c. The lack of cohesion of methods (LCOM) of a class c is the difference between the number of method pairs of c that do not share an instance variable of c and the method pairs of c that do share an instance variable of c. The difference is cut off at zero to prevent negative values. The last two metrics do not belong to the Chidamber and Kemerer metric suite. They are the number of assignments (Assign) and the number of byte code statements (Size) of a class.

4.2 The Projects from the Metrics' point of view

Table 3 presents the metric values for the TDD-projects and the conventional projects. The table lists the minium, the median (med), the maximum, and the mean ($\overline{\mathbf{x}}$). We used the two-sided Wilcoxon test [14, pp. 106] to look for differences in the data samples. The last column of Table 3 shows the p-values. Values smaller than the 5 percent significant threshold are marked. The Wilcoxon test shows a difference for all but two metrics: the depth in the inheritance tree (DIT) and the weighted method per class.

4.3 Assignment Controllability on Method Level

Here, we focus on the values of the assignment controllability on method level. Figure 2 which is located at the end of the paper shows for each project the distribution of the assignment controllability. Two characteristics can be seen. First, most methods have a value for the assignment controllability of 0 or 1.

Table 3. Metric values for the projects.

	conventional				TDD				Wilcoxon
Metric	**min**	**med**	**max**	$\bar{x}$	**min**	**med**	**max**	$\bar{x}$	**p-Value**
AC	0	0.42	1	0.45	0	0.51	1	0.54	<**0.01**
LCOM	0	0	741	3.37	0	1	325	7.28	<**0.01**
RFC	1	10	197	14.35	1	8	91	11.26	<**0.01**
CBO	2	8	165	10.81	2	6	60	8.5	<**0.01**
DIT	1	2	11	2.15	1	1.5	10	2.16	0.32
NOC	0	0	52	0.35	0	0	31	0.48	<**0.01**
WMC	1	5	133	8.29	1	4	59	6.95	0.6
Assign	0	5	273	14.11	0	3	239	6.95	<**0.01**
Size	2	67	2178	140.18	3	57	1762	90.65	<**0.01**

This means that each project has a large number of methods most of which either do not contain any controllable assignment (AC=0, left most bar in each histogram) or in which all assignments are controllable (AC=1, right most bar). A second characteristic is the height of the two bars. Each conventional project has more methods without any controllable assignment than methods in which all assignments can be controlled. This observation holds for Webtest as well, but not for the other TDD-projects. To investigate this topic further, we look at the figures presented in Table 4. It lists for each project the number of methods with at least one non-controllable assignments (AC<1) and the number the methods in which all assignments are controllable (AC=1). We look at the

Table 4. Percentages of controllable methods per project and project group.

	Methods with				
	AC < 1		**AC = 1**		
Project	**number**	**%**	**number**	**%**	**sum**
Webtest	353	58.3	253	41.7	606
XPChess1	46	53.5	40	46.5	86
XPChess2	39	50.6	38	49.4	77
XPChess3	52	47.7	57	52.3	109
Yaps	138	59.0	96	41.0	234
TDD	628	56.5	484	43.5	1112
Ant	1144	63.1	669	36.9	1813
JUnit	146	68.9	66	31.1	212
Log4j	619	73.0	229	27.0	848
conv	1909	66.4	964	33.6	2873
all	2537	63.7	1448	36.3	3985

TDD-projects. Here, 43.5 percent of all methods have assignments which are

completely controllable. See the fourth value in the row labelled TDD. The conventional projects achieve a value of 33.6 percent. The fraction of methods where all assignments are controllable to methods where at least one assignment is not controllable is $484/628 = 0.771$ for the TDD-projects. The fraction for the conventional projects is $964/1909 = 0.505$. The fraction for the conventional projects is smaller than for the TDD-projects. The fraction for the whole data set is $1448/2537 = 0.571$. Finally, the fraction for the TDD-projects is $0.771/0.505 = 1.526$ times larger than for the conventional projects.

4.4 Correlation Analysis on Class Level

This section analyses the correlation of the assignment controllability to the other metrics used in this study. Correlation analysis was performed using Spearman's method. Table 5 shows the correlation coefficients for the corresponding data sets. The column labelled *all* shows the results for the pooled data set. The

Table 5. Correlation analysis on class level.

	AC		
	all	**TDD**	**conv.**
Assign	-0.30	-0.19	-0.32
Size	-0.34	-0.32	-0.35
WMC	-0.20	-0.19	-0.26
DIT	-0.22	-0.26	-0.20
NOC	-0.07	-0.22	0.02
CBO	-0.27	-0.20	-0.30
RFC	-0.32	-0.31	-0.32
LCOM	-0.23	-0.21	-0.29

following columns list the results for the TDD-projects and the conventional projects, respectively. Two effects can be seen. First, all absolute values are smaller or equal 0.35. These small values indicate a low correlation and it seems as if assignment controllability covers a property which is not covered by the other metrics analysed in this paper. And second, there is a negative correlation of the assignment controllability to all other metrics for the *all* and the *TDD* data sets. The negative correlation of the size metric means for example, that small classes tend to have more controllable assignments in their methods than large classes. A similar statement holds for classes with a small number of assignments (Assign), for classes with a small depth of inheritance (DIT), and for classes with low coupling (CBO). It seems as if the assignment controllability metric supports the rules of thumb for testable code.

4.5 Logistic Regression

The applicability of the assignment controllability as indicator for the usage of test-driven development is analysed. Logistic regression is used for this analysis

[15]. Logistic regression is an extension of linear regression to values on a nominal scale. The type of the project is coded by a binary variable. All classes from projects developed with test-driven development are coded with TDD=1. The remaining classes are coded with TDD=0. The logistic model is as follows:

$$P(TDD = 1|X_1, \ldots, X_9) = \frac{1}{1 + e^{-f(X_1,\ldots,X_9)}}$$

$$f(X_1, \ldots, X_9) = \alpha + \sum_{i=1}^{9} \beta_i X_i$$

The enhance readability, the variables X_i $(i = 1, \ldots, 9)$ represent the metrics used in this study. We are looking for parameter values with whom we can estimate the probability whether a project was developed with test-driven development or not. We are not interested in the actual values of α and the β_i. We would rather like to know which metric plays a role in the model and how large its impact on this model is. We estimate the parameters (β_i and α) for two data sets. The data set D_{All} contains all classes while the data set $D_{Assign>0}$ contains only those classes containing at least one assignment.

Table 6 lists for each data set the estimated parameter values and the corresponding standard error. The p-values in the last column refer to the hypothesis test that the parameter has no impact on the model. These p-values are interesting for this analysis. Only α and the assignment controllability have an

Table 6. Logistic model parameter estimates.

	$\mathbf{D_{All}}$			$\mathbf{D_{Assign>0}}$		
Parameter	**Estimated**	**Std. Err.**	**p-Value**	**Estimated**	**Std. Err.**	**p-Value**
α	-1.1955	0.2042	<**0.001**	-0.7337	0.2129	<**0.001**
AC	0.7171	0.2136	<**0.001**	0.8884	0.2306	<**0.001**
Assign	-0.0393	0.0123	**0.001**	1.9888	62.9964	0.974
Size	0.0013	0.0013	0.312	-1.9909	62.9964	0.974
WMC	0.0134	0.0160	0.402	0.0159	0.0161	0.324
DIT	0.0609	0.0503	0.225	-0.0034	0.0533	0.949
NOC	0.0245	0.0294	0.404	0.0246	0.0308	0.424
CBO	-0.0006	0.0209	0.975	0.0163	0.0226	0.468
RFC	0.0082	0.0201	0.683	-0.0058	0.0203	0.773
LCOM	0.0070	0.0051	0.169	0.0069	0.0051	0.180

impact on the model for both data sets ($p<0.001$). The number of assignments is significant for the D_{All} data set as well. All other p-values are larger than 10 percent. Looking at the classes with at least one assignment our data set suggests that the assignment controllability metric is a better indicator for the usage of test-driven development than all the other metrics used in this paper.

4.6 Validity

There are two major threats concerning the validity of the results. First, the data set of the TDD-projects is smaller than the data set of the conventional projects. The main reason for this difference was the absence of industrial TDD-projects. To overcome this shortcoming, we added the three student projects to our analysis. Adding the student projects to the analysis increases the data set. But now, we have three projects from the same problem domain. However, the three projects have been developed by different student groups.

The next problem originates from the usage of student projects. It is unclear how projects developed by developers experienced in test-driven development differ from projects which have been developed by developers new to test-driven development. Students have problems getting accustomed to the test-driven development process [16, 17]. But whether their program code differs from that written by professional developers is not known so far. Thus, the shown differences might not only be caused by the usage of test-driven development but also by the differences caused by the usage of projects developed by students.

5 Conclusions

This paper investigated the assignment controllability of methods. We compared projects which have been developed using test-driven development to conventional projects. Our data set supports the following results:

- The number of methods where all assignments are completely controllable is higher for projects developed with test-driven development than for conventional projects.
- The metric assignment controllability is negatively correlated to all other metrics studied in this paper. The negative correlation supports the rule of thumb of testable programs.
- Assignment controllability is the only parameter that has a significant impact on the predictability whether a project was developed with test-driven development or not.

This study is a first step towards an understanding of the effects of test-driven development on the program code. Further studies should repeat this analysis with a larger data set to increase the validity of the results. Other metrics should be incorporated into the analysis as well, such as complexity metrics or coverage measures of existing tests.

6 Acknowledgement

I would like to thank Guido Malpohl for proof reading a previous version of this paper and Christian Frommeyer for implementing the assignment controllability calculator.

References

1. Müller, M., Hagner, O.: Experiment about test-first programming. IEE Proceedings Software **149**(5) (2002) 131–136
2. Pancur, M., Ciglaric, M., Trampus, M., Vidmar, T.: Towards empirical evaluation of test-driven development in a university environment. In: EUROCON 2003. Computer as a Tool. The IEEE Region 8. Volume 2. (2003) 83–86
3. George, B., Williams, L.: An initial investigation of test driven development in industry. In: ACM symposium on Applied computing, Melbourne, Florida, USA (2003) 1135–1139
4. Geras, A., Smith, M., Miller, J.: A prototype empirical evaluation of test driven development. In: International Symposium on Software Metrics (Metrics), Chicago, Illinois, USA (2004) 405–416
5. Erdogmus, H., Morisio, M., Torchiano, M.: On the effectiveness of the test-first approach to programming. IEEE Transactions on Software Engineering **31**(3) (2005) 226–237
6. Beck, K.: Aim, fire. IEEE Software **18**(5) (2001) 87–89
7. Binder, R.: Design for testability in object-oriented systems. Communications of the ACM **37**(9) (1994) 87–101
8. Chidamber, S., Kemerer, C.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering **20**(6) (1994) 476–493
9. : Digital Systems Testing and Testable Design. Computer Science Press (1990)
10. Apache: Byte code engineering library (BCEL). (http://jakarta.apache.org /bcel/index.html)
11. Canoo: Webtest. (http://webtest.canoo.com)
12. Apache: Ant. (http://ant.apache.org/)
13. Apache: Log4j. (http://logging.apache.org/)
14. Hollander, M., Wolfe, D.: Noparametric Statistical Methods. 2nd edn. John Wiley & Sons (1999)
15. Kleinbaum, D.: Logistic regression: a self-learning text. Springer (94)
16. Wilson, D.: Teaching xp: A case study. In: XP Universe, Raleigh, NC, USA (2001)
17. Müller, M., Link, J., Sand, R., Malpohl, G.: Extreme programming in curriculum: Experiences from academia and industry. In: Conference on Extreme Programming and Agile Processes in Software Engineering (XP2004), Garmisch-Partenkirchen, Germany (2004) 294–302

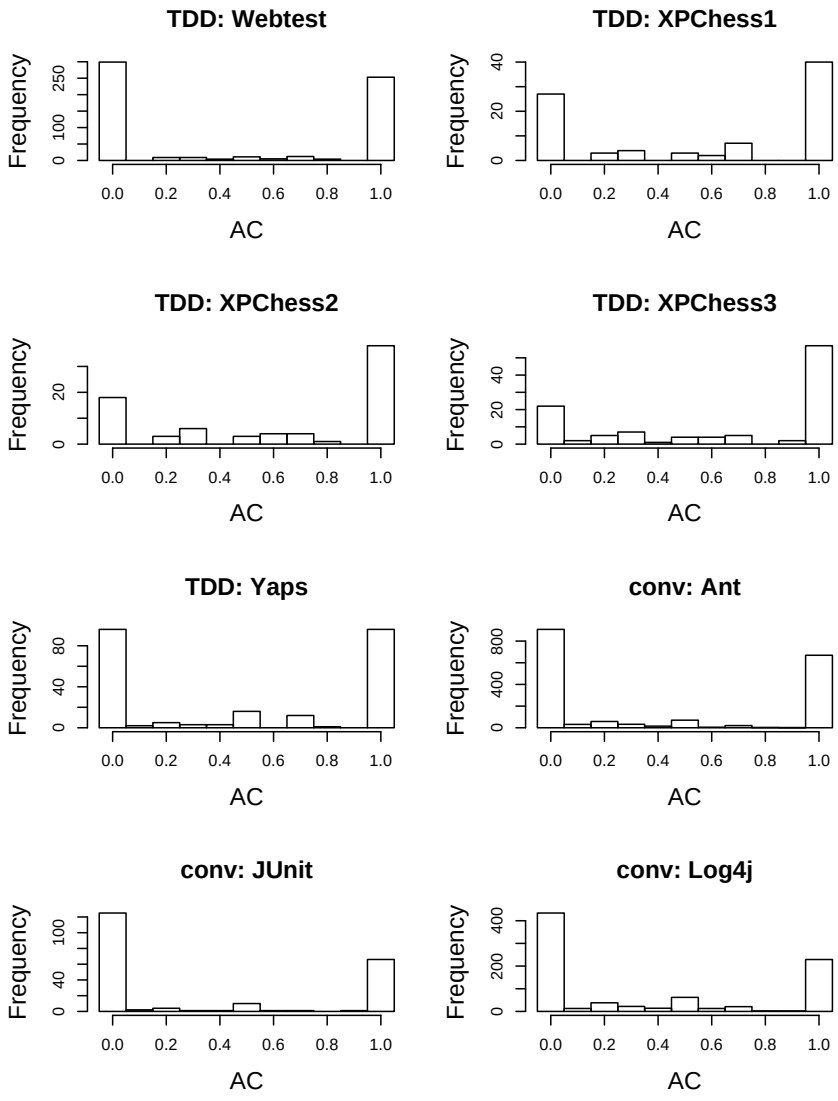

Fig. 2. Distribution of AC on method level for all projects.