About the Return on Investment of Test-Driven Development

Matthias M. Müller
Fakultät für Informatik
Universität Karlsruhe, Germany
muellerm@ira.uka.de

Frank Padberg
Fakultät für Informatik
Universität Karlsruhe, Germany
padberg@ira.uka.de

Abstract

Test-driven development is one of the central techniques of Extreme Programming. However, the impact of test-driven development on the business value of a project has not been studied so far. We present an economic model for the return on investment when using test-driven development instead of the conventional development process. Two factors contribute to the return on investment of test-driven development: the productivity difference between test-driven development, and the conventional process and the ability of test-driven development to deliver higher quality code. Furthermore, we can identify when TDD breaks even with conventional development.

1 Introduction

Test-driven development (TDD) is the only way of coding in Extreme Programming (XP). TDD is also known as test-first programming: write down a simple test for each small piece of functionality before you start coding the functionality. TDD guides you through the whole life-cycle of an XP project. There is no design and no explicit testing phase. Both are replaced by automated tests which are executed continuously to ensure high program quality.
Proponents of TDD claim that it leads to faster development and to more reliable code. Both properties would make TDD superior to the conventional development style which is comprised of a detailed design, a coding phase, and test. First empirical evidence shows [2] though that the claim of faster development might not hold in general; even worse, the opposite seems to be true. Therefore, in order to assess TDD we must study the tradeoff between a (possibly) increased development cost for TDD versus a corresponding gain in code quality.
In this paper, we present an economic model for the return on investment of TDD based on the following two assumptions.

- The development with TDD is slower.
- TDD leads to higher quality code.

Other aspects of TDD, e.g. the cost of continuous testing, are not captured explicitly by our model as their impact on the monetary value of the project can not be easily separated. Thus, we consider our model as a first major step towards a full economic assessment of TDD, and it adds to the description of the economic benefit of XP projects [3].
The model compares the development cost for a conventional project with the development cost for a project that uses TDD. The investment cost is the additional effort necessary to complete the TDD project as compared to the conventional project. The life cycle benefit is captured by the difference in quality measured by the number of defects that the TDD team finds and fixes, but the conventional project does not. This defect difference is transformed into a monetary value using the additional developer effort corresponding to finding and fixing these defects in the conventional project. The concepts of the life cycle benefit and the investment cost in our context are depicted in figure 1. The upper horizontal line corresponds to the conventional project with additional quality assurance phase! The lower horizontal line corresponds to the TDD project.
Our model captures the return on investment for an experienced TDD team. Additional cost for training necessary when introducing TDD is not considered.
With this model, we can identify tradeoff lines where TDD becomes beneficial over conventional development. Interestingly, the break-even point is independent of the actual project size, the number of developers per team, and the actual developer salary; the decisive data are productivity difference, quality difference, defect removal time for one defect, working time per developer per month, and the initial defect density.

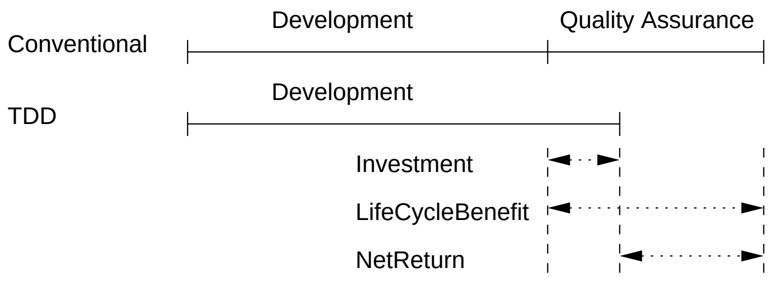

Figure 1. Overview of benefit cost ratio calculation.

2 Model

This section describes those formulas of our model which are necessary to understand the break-even analysis in Section 3. Appendix A contains a comprehensive description of the model formulas.

2.1 Return on Investment

Calculating the return on investment ROI means to add up all the benefits of the investment, subtract the cost, and then compute the ratio of the cost:

$$\mathsf{ROI} = \frac{\mathsf{LifeCycleBenefit} - \mathsf{Investment}}{\mathsf{Investment}}$$

If the investment pays off, the ROI is positive, otherwise negative. In our evaluation of TDD we focus on the benefit cost ratio BCR which is easily derived from the return on investment.

$$\begin{aligned}\mathsf{BCR} &= \frac{\mathsf{LifeCycleBenefit}}{\mathsf{Investment}}\\ &= \mathsf{ROI} + 1\end{aligned}$$

Studying the BCR instead of the ROI makes the break-even analysis much simpler, see below.

2.2 Investment Cost

We first look at the *investment* cost. For the conventional project, the development phase includes design, implementation and test. The development phase of the TDD project is comprised only of test-driven development.
As first empirical evidence suggests, we assume that the TDD project lasts longer than the conventional project. We call the ratio of the project durations the *test-speed-disadvantage* (TSD).

$$\mathsf{TSD} = \frac{\mathsf{Time}_{\mathsf{Conv}}}{\mathsf{Time}_{\mathsf{TDD}}}.$$

Since we assume that the development phase is shorter for the conventional project, the test-speed-disadvantage ranges between $\mathsf{0}$ and $\mathsf{1}$:

$$0 < \mathsf{TSD} < 1.$$

Using productivity figures to explain the difference in elapsed development time between the two kinds of project, the TDD development is $(1 - \mathsf{TSD}) \times 100\,\%$ less productive than the conventional project.
Finally, the investment is the difference between the development cost of the TDD project and the conventional project.

2.3 Life Cycle Benefit

Now, we consider the *benefit.* Each development process is characterized by a distinct *defect-removal-efficiency* (DRE). The defect-removal-efficiency denotes the percentage of defects a developer eliminates during development. Initially, a developer inserts a fixed amount of defects per thousands lines of code (*initial-defect-density*, IDD), but he eliminates $\mathsf{DRE} \times 100\,\%$ of the defects during the development process. From the increased reliability assumed for TDD, we have

$$0 < \mathsf{DRE}_{\mathsf{Conv}} < \mathsf{DRE}_{\mathsf{TDD}} < 1.$$

The additional *quality assurance* (QA) phase of the conventional project compensates for the reduced defect-removal-efficiency of the conventional process. The only purpose of the QA phase is to remove all those defects found by TDD but not by the conventional process. The amount of defects to be removed in the QA phase is mainly characterized by

$$\triangle\mathsf{DRE} = \mathsf{DRE}_{\mathsf{TDD}} - \mathsf{DRE}_{\mathsf{Conv}}.$$

The benefit of TDD is equal to the cost of the QA phase for the conventional project. The benefit depends on the effort (measured in developer months) for repairing one line of code during QA, which is characterized by

$$\mathsf{QAEffort} = \frac{\mathsf{DRT} \times \mathsf{IDD}}{\mathsf{WT}}$$

QAEffort depends on the following:

- The defect removal time DRT. It describes the developer effort in hours for finding and removing one defect.
- The inital defect density IDD. The number of defects per line of code inserted during development.
- The working time WT. The working hours per month of a developer.

The reciprocal of QAEffort is a measure for the productivity during the QA phase.

2.4 Benefit Cost Ratio

The benefit cost ratio is the ratio of the benefit and the investment. Substituting the detailed formulas of our model given in Appendix A, the benefit cost ratio becomes

$$\mathsf{BCR} \;=\; \mathsf{QAEffort} \times \mathsf{Prod} \times \frac{\triangle\mathsf{DRE} \times \mathsf{TSD}}{(\,1 \,-\, \mathsf{TSD}\,)}, \qquad (1)$$

where Prod is the productivity of the conventional project during the development phase measured in lines of code per month. Values larger than 1 for the BCR mean a monetary gain from TDD, values smaller than 1 a loss.

2.5 Break Even

Setting the benefit cost ratio equal to 1, we get a relation between the test-speed-disadvantage of TDD and the reliability gain of TDD:

$$\mathsf{TSD} \;=\frac{1}{\mathsf{c} \,\times\, \triangle\mathsf{DRE} \,+\, 1}, \text{ or}$$

$$\triangle\mathsf{DRE} \;=\frac{1 - \mathsf{TSD}}{c \times \mathsf{TSD}}$$

$$c = \mathsf{QAEffort} \times \mathsf{Prod}$$

This relation characterizes the break-even point for TDD. If the difference between the defect-removal-efficiencies is known, a lower bound for the test-speed-disadvantage can be calculated from which on the TDD project starts to be beneficial.

3 Results

3.1 Exploring the Benefit Cost Ratio

As an example, we examine the benefit cost ratio of the following scenario.

Factor	Value
DRT	10 h/defect
IDD	0.1 defects/LOC
WT	135 h/month
Prod	350 LOC/month

Let TSD and △DRE vary. Figure 2 shows the benefit cost ratio plane spanned by the test-speed-disadvantage TSD and the defect-removal-efficiency difference △DRE. Values larger than 4 are cut off.

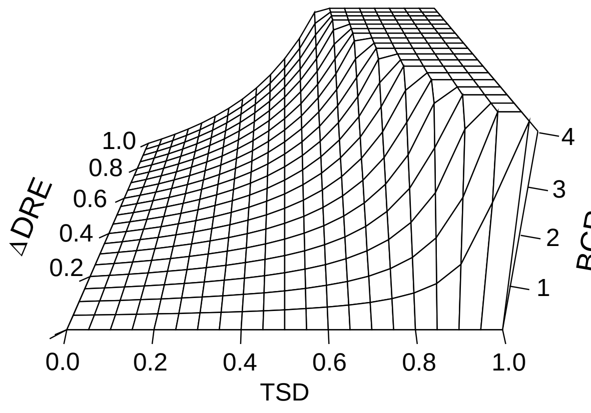

Figure 2. Benefit cost ratio dependent on TSD **and** △Eff

For large values of the test-speed-disadvantage ($\mathsf{TSD} > 0.9$) the TDD project performs almost always better than the conventional project, even for a small defect-removal-efficiency difference. On the other hand, if the test-speed-disadvantage is very small ($\mathsf{TSD} < 0.2$), TDD does not produce any benefit regardless how large the defect-removal-efficiency difference is.
The following table shows some benefit cost ratios for selected values of TSD and △DRE.

TSD = 0.9

△DRE	BCR
0.01	1.0 : 4.3
0.05	1.7 : 1
0.1	2.3 : 1

TSD = 0.3

△DRE	BCR
0.2	1 : 4.5
0.4	1 : 2.3
0.6	1 : 1.5
0.8	1 : 1.1
0.9	1 : 1

If the productivity of TDD is 10 % smaller than the productivity of the conventional project (left table), a 5 % better defect-reduction-efficiency suffices for TDD to break-even with the conventional process (1.7 : 1). If the productivity of TDD is much worse, say, 70 % smaller (right table), even a 80 % better defect-reduction-efficiency does not lead to a gain as compared to the conventional process (BCR is 1 : 1.1).

3.2 Break Even Analysis

With break even analysis, the ranges for TSD and $\triangle$DRE can be identified where TDD is more beneficial than the conventional process. Figure 3 shows the intersection of the surface in figure 2 with the horizontal plane BCR $= 1$.

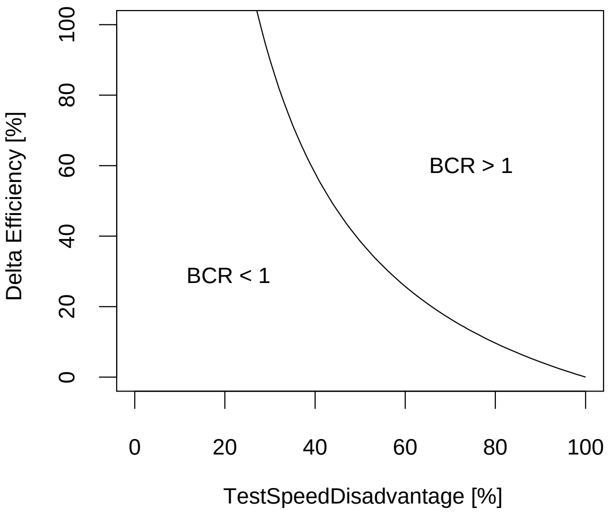

Figure 3. TSD **and** $\triangle$Eff **plane for** BCR $= 1$

The right half of the plane (BCR > 1) corresponds to the parameter range of TSD and $\triangle$DRE where TDD is superior over conventional development. Two observations can be made. First, assuming that practical values for $\triangle$DRE can not be larger than 20 %, the TSD may not drop below 66 % for TDD, otherwise the TDD cost exceeds its benefit. Second, if the TSD drops below 27 %, TDD does not have a chance to provide any financial return, regardless of how large the improved defect-removal-efficiency may be.

3.3 Varying other project parameters

Figure 4 shows the different cost benefit break-even lines for varying values of the programmer productivity Prod. All other parameters are kept constant.

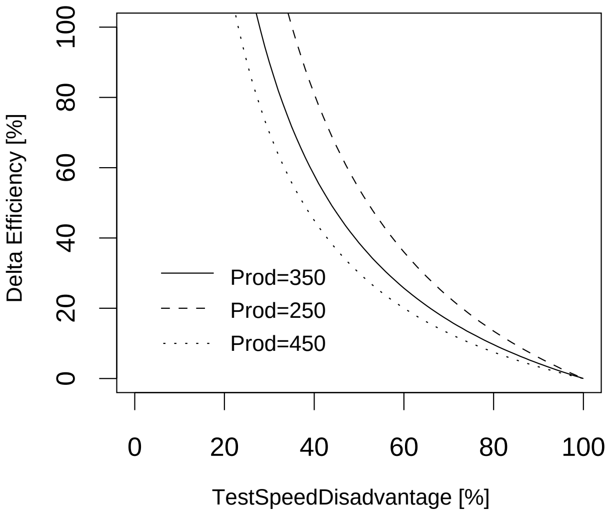

Figure 4. Break even analysis for varying values for Prod

The higher the initial productivity, the higher the chance for TDD to get a financial return over conventional development. This result is not intuitively obvious but, it can easily be derived from (1) and explained with figure 1 as follows. The higher the productivity the shorter the elapsed development time for both TDD and the conventional project. If the elapsed time for the development phase decreases, the investment (difference between both development phases) also decreases, and thus the benefit cost ratio becomes larger.

4 Conclusions

We propose an economic model for the return on investment of test-driven development. Our analysis of the break-even leads, all other parameters are kept constant, to the following conclusions:

- The return on investment of TDD depends to a large extend on the slower development of TDD and the higher quality code of TDD.
- Other factors like the effort for fixing a faulty line of code, or, the productivity of a developer using the conventional development process, have only minor impact on the return on investment of TDD.
- The calculation of the return on investment is independent of the project size, the number of developers, and the developer salary.

Our model assumes an experienced TDD team. The additional cost for training which is necessary when first introducing TDD is ignored so far.

Finally, our model strengthens the need for actual empirical figures (or ranges) for the quality advantage and the loss of productivity of TDD, in order to get a comprehensive evaluation of the cost and benefit of TDD.

References

[1] W. Humphrey. *A discipline for software engineering.* Addison-Wesley, 1997.

[2] M. Müller and O. Hagner. Experiment about test-first programming. *IEE Proceedings Software*, 149(5):131–136, Oct. 2002.

[3] M. Müller and F. Padberg. Extreme programming from an engineering economics point of view. In *International Workshop on Economics-Driven Software Engineering Research (EDSER)*, Orlando, Florida, May 2002.

[4] I. Sommerville. *Software Engineering.* Addison-Wesley, 1995.

A Appendix

A.1 Factors in the Economic Model

The following list explains the factors and their abbreviations used throughout the model.

ProductSize Size of the project in lines of code.

Prod The developer productivity measured in lines of code written per month. This figure includes design, coding, and testing. We assume that the productivity remains constant for all developers during the project. The average productivity ranges between 250 and 550 lines of code per month [4].

Salary Salary for the whole team per year. This factor is not further broken down as it turns out that our model is independent from the actual value for the salary.

NumOfDev The number of developers working in the project. We assume that this number is fixed throughout the whole project.

$\mathsf{DRE}_{\mathsf{TDD}}$/ $\mathsf{DRE}_{\mathsf{Conv}}$ The defect removal efficiency describes the percentage of defects a team removes during development. We assume that TDD has a higher defect removal efficiency than the conventional process.

TSD The test-speed-disadvantage accounts for the additional effort for using TDD during development as compared to the conventional process.

DRT The defect removal time denotes the effort for finding and removing one defect during the QA phase measured in hours per defect.

IDD The number of defects per thousand lines of code inserted during development is described by the initial defect density. A typical number is 100 defects per thousands lines of code [1]. A developer reduces this number of defects according to his defect removal efficiency.

WT The working hours of a developer each month.

A.2 Model Formulas

For the conventional project, the development time is

$$\mathsf{Time}_{\mathsf{Conv}} = \frac{1}{12} \times \frac{\mathsf{ProductSize}}{\mathsf{Prod} \times \mathsf{NumOfDev}}.$$

For the TDD project, the decreased productivity has to be taken into account:

$$\mathsf{Time}_{\mathsf{TDD}} = \frac{\mathsf{Time}_{\mathsf{Conv}}}{\mathsf{TSD}}$$

During the QA phase, the conventional project has to compensate for the lower defect removal efficiency as compared to TDD:

$$\triangle\mathsf{DRE} = \mathsf{DRE}_{\mathsf{TDD}} - \mathsf{DRE}_{\mathsf{Conv}}$$

There have to be

$$\triangle\mathsf{Defect} = \mathsf{ProductSize} \times \mathsf{IDD} \times \triangle\mathsf{DRE}$$

defects removed during QA to get the same defect density as the TDD project. Thus, the time spent in the QA phase is

$$\mathsf{Time}_{\mathsf{QA}} = \frac{1}{12} \times \frac{\mathsf{DRT} \times \triangle\mathsf{Defect}}{\mathsf{WT} \times \mathsf{NumOfDev}}$$

The cost for both the TDD and the conventional project and the QA phase is

$$\mathsf{Cost}_{\mathsf{p}} = \mathsf{Time}_{\mathsf{p}} \times \mathsf{Salary}$$

where $\mathsf{p} \in \{\mathsf{TDD}, \mathsf{Conv}, \mathsf{QA}\}$.

A.3 Calculating the BCR

The benefit cost ratio is defined as the ratio between the life cycle benefit and the investment (cost):

$$\mathsf{BCR} = \frac{\mathsf{LifeCycleBenefit}}{\mathsf{Investment}}$$

Recall that the life cycle benefit equals the cost for the additional QA phase in the conventional process.

$$\mathsf{BCR} = \frac{\mathsf{Cost}_{\mathsf{QA}}}{\mathsf{Cost}_{\mathsf{TDD}} - \mathsf{Cost}_{\mathsf{Conv}}}$$

The factor Salary can be canceled out. Hence,

$$\begin{aligned}\mathsf{BCR} &= \frac{\mathsf{Time}_{\mathsf{QA}}}{\mathsf{Time}_{\mathsf{TDD}} - \mathsf{Time}_{\mathsf{Conv}}} \\ &= \frac{\mathsf{Time}_{\mathsf{QA}}}{\mathsf{Time}_{\mathsf{Conv}} \times (\frac{1}{\mathsf{TSD}} - 1)}.\end{aligned}$$

Further canceling of the factors $\mathsf{12}$, $\mathsf{NumOfDev}$, and $\mathsf{ProductSize}$ leads to

$$\begin{aligned}\mathsf{BCR} &= \frac{\mathsf{DRT} \times \mathsf{IDD} \times \mathsf{Prod} \times \triangle\mathsf{DRE}}{\mathsf{WT} \times (\frac{1}{\mathsf{TSD}} - 1)} \\ &= \frac{\mathsf{DRT} \times \mathsf{IDD} \times \mathsf{Prod} \times \triangle\mathsf{DRE} \times \mathsf{TSD}}{\mathsf{WT} \times (1 - \mathsf{TSD})} \\ &= \frac{\mathsf{DRT} \times \mathsf{IDD}}{\mathsf{WT}} \times \mathsf{Prod} \times \frac{\triangle\mathsf{DRE} \times \mathsf{TSD}}{(1 - \mathsf{TSD})} \\ &= \mathsf{QAEffort} \times \mathsf{Prod} \times \frac{\triangle\mathsf{DRE} \times \mathsf{TSD}}{(1 - \mathsf{TSD})}.\end{aligned}$$