

**Masters Thesis Software Engineering**

# Developing a suitable structured Testing Approach for a small web development company

A Case study at Basket Builders BV



UNIVERSITEIT VAN AMSTERDAM



## Students

**Peter R. Lamers**  
Goudwesmeent 53  
1218 GP Hilversum, The Netherlands  
+31 6 27 205 064  
[prlamers@hotmail.com](mailto:prlamers@hotmail.com)

**August L. P. de l'Annee de Betrancourt**  
Straatsburgflat 18A  
1422 VV Uithoorn, The Netherlands  
+31 297 527888 / +31 6 51609516  
[august.lannee@xs4all.nl](mailto:august.lannee@xs4all.nl)

## Company

**Basket Builders BV**  
Prinseneiland 7  
1013 LL Amsterdam  
+31 20 42 22 383  
[info@b-b.nl](mailto:info@b-b.nl)

Final Assignment paper

Version 1.0

June 24 2004

Students:

Peter R. Lamers & August L. P. de l'Annee de Betrancourt

Mentor (Basket Builders):

David Smits ([david@b-b.nl](mailto:david@b-b.nl))

Mentor (UvA):

Alban Ponse ([alban@science.uva.nl](mailto:alban@science.uva.nl))

## PREFACE

This document is our Master Project Thesis for the study “Software Engineering” at the University of Amsterdam.

The title of our Master Project assignment is “Developing a suitable structured testing approach for a small web development company”.

We chose to do this project as a cooperation between 2 students because of the limited time available (3 months). Furthermore, we believe that through combined viewpoints one can obtain a higher quality level.

The subject of software testing was given a reasonable amount of attention within our study. For the subject Software Testing we studied the TMap theory, which is quite a formal and extensive testing method. Also for the subject Software Process the testing workflow within the overall process was given a considerable amount of attention. It is interesting to see how testing is handled in practice: which problems surround the testing workflow? Which theory on testing is applicable? And how does the testing activity fit into the total development process?

We executed this project for and at Basket Builders BV, a web application development company.

We experienced it as very useful and because of the interesting project definition, we were able to perform a research in which we gained a lot of knowledge and could apply much of the theory learned during our study.

Hereby, we would like to thank our counselors, David Smits and Bart Ferwerda (from Basket Builders) and Alban Ponse (from UvA) whom continuously provided us with good and critical advice and guidance.

Furthermore, we would like to thank: Jos van Rooyen (LogicaCMG), Martin Pol (founder TMap), Reza Esmaili, Mark van den Brand, Hans ter Wal and Justin van Beijereren for their help during the project.

We composed this Thesis in the form of a paper. It is divided into 8 sections. Section I and II are introductory and describe the company, the research questions and give a concise overview of the followed

approach. In order to get straight to the essence of the matter, this part can be passed over and one can directly begin with Section III, which describes the current situation concerning testing at Basket Builders.

Part IV holds an overview of the results of our extensive literature study after which in part V the answers to the research questions and the recommendations following from the study are described.

These recommendations were tested in a project and the findings are described in Section VI.

The Thesis is concluded with a conclusion in Section VII, an overview of the references (Section VIII) and, a number of attachments.

*"If you think good testing is expensive, try bad testing."*

*(Andres Villavicencio)<sup>1</sup>*

---

<sup>1</sup> from <http://www.testen.nl>

## INDEX

PREFACE .....	2
INDEX.....	2
INDEX.....	3
Abstract.....	4
Keywords.....	5
SECTION I: BACKGROUND.....	6
The company.....	6
Project motivation.....	6
Research question .....	6
Goals.....	6
Scope.....	6
SECTION II: RESEARCH APPROACH.....	7
SECTION III: CURRENT SITUATION.....	8
Current project approach.....	8
Current issues.....	9
The *Net Toolbox.....	10
Requirements .....	10
SECTION IV: LITERATURE STUDY.....	12
Basic principles.....	12
The process .....	12
Available testing methods and techniques .....	14
Supporting tools.....	16
SECTION V: RESULTS.....	17
General.....	17
Recommendations.....	17
Quality assurance.....	20
SECTION VI: THE CASE.....	21
Case description.....	21
Project realization .....	21
Findings: evaluation.....	21
SECTION VII: CONCLUSION.....	22
Glossary of terms .....	22
SECTION VIII: REFERENCES .....	24
Literature references .....	24
Software references.....	25
POSTFACE.....	26
ATTACHMENT A: Screenshots of the Poll User Control.....	29
ATTACHMENT B: Summary Testing Manual for Basket Builders.....	30
ATTACHMENT C: TMap Rood.....	32

# Developing a suitable structured Testing Approach for a small web development company

A Case study at Basket Builders BV  
by August de l'Annee de Betrancourt & Peter Lamers

**T**

here is a wide selection of testing methods and techniques available. Some are formal; some leave a lot of room for creativity. Testing is a key activity within the software development process though often a “suppressed matter”. This is remarkable since it typically takes in 25-50% of the total development time.

This especially applies to smaller development projects and companies where finding a balance between structure and flexibility is a constant tradeoff.

Although a small web development company requires an adapted testing approach compared to high risk, business critical projects where formal methods are more directly applicable, a structured testing approach is certainly needed. Combined with a good set of tools, guidelines and support from within the organization (especially from the developers), a new testing approach can be successfully adopted by an organization and integrated into its projects resulting in a higher quality level.

## Abstract

Our case study was done at a web development company called Basket Builders. This is a company that currently does not apply a formal development method in general. The main requirement within this company is a structured testing approach because testing is now mostly done at the end of a project.

After analyzing the development activities and method of the company, we identified the problem areas and requirements concerning testing.

In a literature study we focused on the available theory and solutions on testing smaller non-critical web development projects.

With small projects it is difficult to fit in all the good testing practices without overplaying the testing approach.

Concerning the use of formal testing techniques, the balance between extra (maintenance) costs of test scripts versus the advantage of having standard and conserved formal tests is an important consideration. Furthermore, a lot of test scripts tend to become outdated. The ability

of running them more than once is not always a relevant asset.

One basically wants a test roadmap that:

- provides a structured and risk based guideline to make sure that all the important tests are performed.
- is easy to compose and maintain.
- is dynamic in the sense that it leaves enough room for the tester to use creativity and respond to findings following from the test.

The formal method TMap [19] is too extensive for complete use within small projects of low risk. Exploratory Testing [17] offers more dynamics and less overhead and maintenance.

However TMap can be handled as a toolbox and so it offers several formal techniques which are indeed useful. Formal techniques provide a certain degree of assurance on the test coverage.

It is important that the overall process is structured first. If there are flaws in the requirements and specifications, the testing activity can never restore this.

By studying agile methods as DSDM [29], RUP [30] and Extreme Programming (also referred to

as XP) [10], we defined additional recommendations concerning the overall software approach in cooperation one of the project managers.

The testing approach itself can be separated into a global part and parts that are customized for the different development activities. The recommendations include:

- Use of code reviews for early fault detection (whitebox testing).
- Set up an acceptance site for functional testing by the customer.
- In the development phase, write Unit tests using the NUnit framework [22]. In particular NUnitASP [35].
- Apply Test Driven Development from XP within development of the Content Management System (CMS).
- Apply at least the Dataflow test technique from TMap for formal tests for the CMS.
- Assign a dedicated test manager / coordinator. This may seem expensive and obsolete for a small organization but the coordination and responsibility is centralized. This provides a form of Quality assurance for the testing activity.

The result of the project was a structured testing approach combined with a set of test tools customized for use within Basket Builders' projects.

The new approach now needs to be evaluated within a project at Basket Builders. There was insufficient time to do this during the 3 month assignment. We did however assess the technical part of using NUnit. This was tested in a small project.

### **Keywords**

Unit testing, functional testing, development method, NUnit, software process, Microsoft .NET, testing methods, TMap, Exploratory Testing, Test Driven Development

## SECTION I: BACKGROUND

### The company

The project has been executed at Basket Builders BV. This Amsterdam based company was founded at the end of 1996 and it currently has 26 employees.

Basket Builders develops its own Content Management System (CMS) for web applications, the \*Net Toolbox. Next to developing this system, the company implements their own CMS in website development projects. These projects vary from small to middle size and normally do not exceed a 6 month development time.

A few of Basket Builders' clients are: Neckerman, Sky Radio and Interpolis.

For more information on Basket Builders we refer to the website: <http://www.b-b.nl>.

### Project motivation

Both from the fields of management and development there was a call for a testing approach. Testing is an activity that gets too little attention and above all is now not structured into the development cycle. Testing is mostly done at the end of a project.

As a result there are problems of exceeding time and budget because of program errors that are expensive to fix and late additional requirements. This outlet of projects must be reduced because this will increase the total profit. For example decreasing the average exceeding time with only 1% will pay off this research because this will result in a theoretical profit increase of €6000 per quarter.<sup>1</sup>

### Research question

The following main research question was set:

*How can testing be structured into the software process for small web projects, in order to achieve a higher quality level?*

In order to answer the main research question, we defined smaller research questions that needed to be answered first:

<sup>1</sup> Based on Basket Builders' financial data

Question A: What is the current project approach?

Question B: How are the products currently tested? What procedures are followed?

Question C: What available testing techniques and methods are suitable for Basket Builders?

Question D: Which aspects of the current approach are useful for the testing approach and which should be replaced or rejected?

Question E: How does testing fit into the rest of the software process?

Question F: Which tools can be used to support the testing activities?

### Goals

Not only was it needed to answer the research questions, it was also needed to set some clear goals that needed to be reached. The research questions form a more theoretical approach, while the goals are more practical. The following goals were set:

- Business Driver: Reducing the average exceeding time per working hour with 2% (€10.000 more profit per quarter).
- The main result/goal of this project: Define a structured testing approach supported by the use of software tools and tailored to the needs and current process of Basket Builders B.V. The approach should be described in a compact and clear manual.
- Get an overview of the current project and testing approach.
- Provide additional recommendations concerning the software process that support the testing workflow.
- Set up a knowledge database containing the theory on testing techniques and methods.

### Scope

The scope of this project encloses all methods, techniques, tools and activities concerning the testing workflow. Testing stretches to other fields like the project approach of the company. This overlap was examined to see what parts can be used to support the testing process.

A risk was that the scope would get too wide because the whole development program needs refinements. This is something we had to be particularly aware of.

## SECTION II: RESEARCH APPROACH

Before the actual research could be done some preparations had to be taken. The short timescale and the relatively large scope demanded a fixed plan. In concurrence with good development practises we used time boxing, frequent project deliverables and several evaluation sessions with both stakeholders and experts to verify that everything was going according to plan. A logbook was used to give detailed information about the work that has been done.

The following activities were completed:

1. Execute a small project to obtain more insight in Basket Builders' situation and attain additional background information that is relevant for the research.
2. Study Basket Builders' available project documentation [1]. This method gives, in a more formal way, insight into the current working methods.
3. Interview several programmers and a manager to discuss their opinions on the current situation and needs for the future. Their view on testing was requested and the results were used for future recommendations. This also promotes support for future plans.
4. Gather and study papers relevant to the research question(s). Findings were reviewed weekly together with a project manager at Basket Builders. Testing is a practical matter but also needs theoretical support to understand its background and its necessity. Papers were also useful because they discuss the practical use of testing. Subjects and methods that look useful for Basket Builders were further examined and translated to a practical solution for the company. Because testing for small projects differs substantially from testing for bigger projects, papers have been searched that specifically focus on testing web applications for smaller companies.

5. Findings from the papers, interviews and project documentation were documented into notes for the Thesis. These notes were used to write the section Literature Study and form the basis for our recommendations.

6. During the research period a knowledge database was used to save all relevant documentation (papers, intern documents, results interviews and meeting notes). Because the research was a combined effort by two students, this approach has been considered very useful.

7. To maintain a certain level of quality, findings were discussed with people not directly involved in the project. These people can give a clean new view on the results.

We also wanted some practical input from the business field. We visited a testing conference<sup>2</sup> where some leading persons were present. Furthermore we discussed the Thesis with the head of the testing department of LogicaCMG, Jos van Rooyen, and Martin Pol, founder of structured testing in Holland and author of several books (for example TMap [18]).

8. Our plan was to apply the recommended approach to testing within a pilot project in order to gather tangible measurements and adjust the approach accordingly. Unfortunately we lacked time to evaluate the new approach within a project.

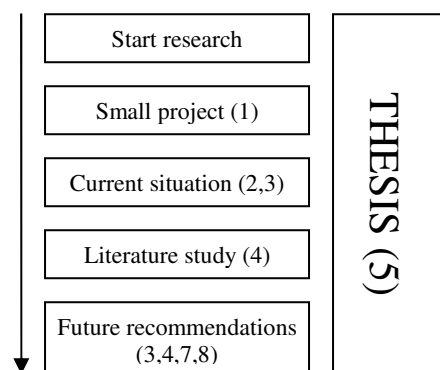


Figure 1: Overview of the research approach

<sup>2</sup> TestNet Spring Event June 9<sup>th</sup> 2004

## SECTION III: CURRENT SITUATION

### Current project approach

In this section Basket Builders' current project approach is documented.

Basket Builders is a small and quite informal company. Projects rarely exceed a period longer than 2 months.

No specific formal development method is applied and a tight project approach is absent. PRINCE2<sup>3</sup> [2] is applied as the project management method. More specifically, the following parts of PRINCE2 are used:

- A Project Initialisation Document is drafted (PID).
- Several other templates of PRINCE2 deliverables are used like the Highlight Report, Project Issue and Request for Change.

Projects are divided into smaller tasks. A projects manager defines and assigns these tasks to a specific developer. These tasks need to be approved first by the head of a team before development can be started.

Basket Builders' project manual [1] describes on a global level how projects need to be executed. The main focus is what roles need to be fulfilled and what stages consist during a project.

The project role hierarchy is divided into 3 levels:

- Project management – acquisition, functional specifications and client contact.
- Team leaders – coordination of development activities and client contact.
- Developers – technical design, pursued clarification on functional specifications and programming.

The development activities can be divided into two main parts:

---

<sup>3</sup> Project IN Controlled Environments [2]

- ✓ Continuous development of Basket Builders' own Content Management System (the code of the CMS is referred to as the Core).
- ✓ Developing web based applications for customers (usually by implementing the \*Net Toolbox).

New developments and changes to the Core are mostly planned and communicated during cluster meetings. There is no fixed quality assurance procedure for maintenance and improvement of the Core.

Client projects normally start as follows. After acquisition and first contacts with the customer, the functional requirements will have to be specified. This is normally done through one of the following methods:

- A functional design is provided by an extern company.
- Workshops are held with customers.

A manager is responsible for acquiring the functional requirements, and provides this information to the head of a group of developers.

After that he will select which developers (based on technique) are going to be deployed for the project. Developers handle further contact with the customer in order to elaborate and refine the functional requirements.

If necessary, a technical design is made. Often, the same technological architecture is used for the project solution. Therefore an extensive technical design is not always needed.

Optionally, a prototype is developed to eliminate any doubt a customer may have. Prototypes are also used to see if a project can be realised (proof of concept).

After the project is completed and handed over, a warrantee stage is initialised. During this stage, a customer can report bugs which are solved. This part can be seen as a testing stage.

At the moment, testing activities are mainly without any structure.

Testing is currently done as follows. After writing a piece of code the developer tests it



himself. Usually the software is also tested by a second person (the head of the project team) though this is not enforced by any procedure. Incidentally, testing scripts are made but normally no testware is developed and conserved because Basket Builders does not apply formal testing techniques.

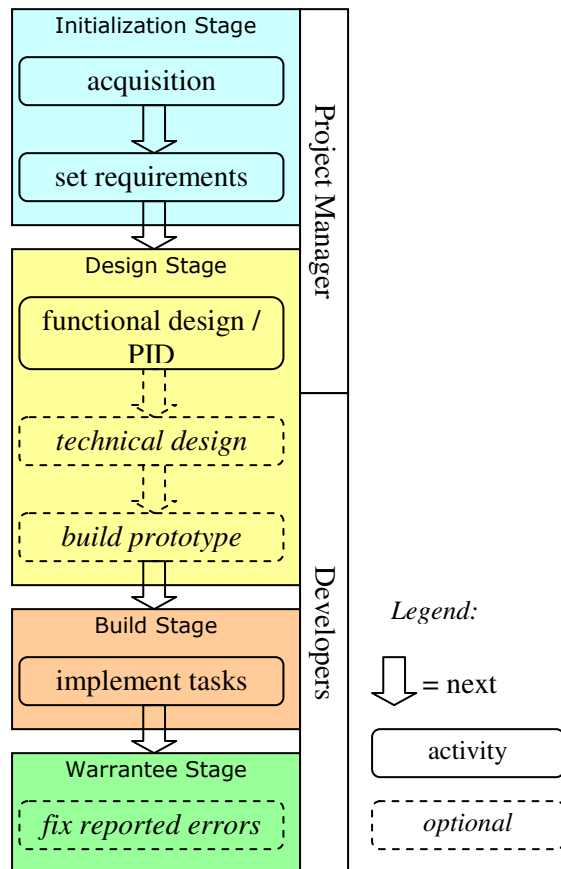


Figure 2: Graphical overview of the process

Unit testing is used but only on a very limited scale: Basket Builders has developed an automatic generator for stored procedures and access functions based on database tables. This application also generates Unit tests that check whether the stored procedures are correctly handled.

For Integration testing [31] a standard checklist is used.

Code reviews have been done in the past but also only incidentally.

## Current issues

When looking at the current development method the following current were found.

### *No structured test approach*

First of all, testing is an activity that is mainly done at the end of a project. Even then there is often too little time available for testing because it is not planned into the project cycle. Tasks are tested by the project manager however.

Last minute functional additions also harm testing. Result is project delay or a product that has many errors. As Basket Builders main objective is satisfying its customers, this is a problem.

### *Projects exceed time schedule*

Projects often exceed the time schedule set at the start of the project. This happens because of changing requirements at the end of a project. A customer does not know exactly what his own needs are. Software is mainly complex, large and overview is missing. Only when the application is ready, a customer can see what can be done with the application.

### *Lack of overview*

During development, team members do not know what their colleagues are doing. An overview of the project is missing. The developers lack awareness of the used project approach, while managers lack information about what the developers are doing exactly.

Managers and developers are standing too far apart from each other.

There are several standard guidelines towards development available but developers are not always aware of this and so they remain unused.

### *\*Net Toolbox complexity*

The Toolbox is quite advanced so it takes more effort to find software errors. Moreover, the Core still contains errors.

Up to date documentation is very limited and flawed which makes it even more complicated.

### *Testing takes too much time*

According to programmers, testing currently takes too much time and effort.

Because testing is mainly done at the end of the project the cost of finding and fixing errors is too high. The further the project progresses, the more time it takes to fix errors and retrofit the necessary design adjustments.

### The \*Net Toolbox

Before discussing the requirements in the next paragraph, a short introduction of the \*Net Toolbox, Basket Builders' Content Management System, is in place.

Extra functionality can be added to the \*Net Toolbox, by using so called *controls* (also called objects in programming terminology). The new functionality can be loaded and configured from the *back-end* and be shown on the *front-end* (the actual website).

*Templates* can be defined to specify a standard design for the web application. *Views* can be defined to show data from a database, on the website.

More info on the \*Net Toolbox can be found on the official website: <http://www.nettoolbox.nl>

### Requirements

Like in every project, even this research, there are requirements that have to be fulfilled. The following requirements were defined in co-operation with the stakeholders:

There is a need for a structured testing approach. This is the main trigger and requirement for this project.

The development activities at Basket Builders can be divided into sub activities. The most important issue is that testing is tuned to match these activities. Some of these activities require special attention and measurements concerning testing.

The following activities need to be considered (when focusing on development) for the testing approach:

- 1 The provided front-end HTML, that will be implemented in the \*Net Toolbox, must be tested to ascertain it will work on different browsers and platforms. HTML can easily be tested by using a tool.

2. So called Templates & Views are used to show content within the front-end. Both need to be tested because they will show the actual information to a web visitor. This needs to be done manually because this concerns the front-end of the CMS.
3. Adding new functionality to the \*Net Toolbox can easily be done by using .NET User Controls. These controls are developed in the .Net development environment and then placed into the \*Net Toolbox. Both Functional and Unit tests are needed (whitebox testing). The User Controls can then be blackbox tested in the CMS.
4. The total integration of Controls, Views and Templates need to be tested. Again a more manual approach is needed because this can only be tested from within the CMS.
5. After installing the CMS on the actual working server the application needs to be tested again. Certain variables need to be set too. A semi-automatic approach is desired here.
6. The CMS is constantly under development. Core development needs a specific test approach.
7. Finally stress testing is needed to verify the application can deal with a big group of users. This can also be done automatically.

When examining the current approach and gathering the opinion of the developers, it is clear to see that there is more need for a 'lightweight, more creative' method than a tight one. The projects are too small for very formal and extensive methods. The projects are not business critical so there is little need for formal methods.

A lightweight method like Extreme Programming is preferred by many people working at Basket Builders. According to them, if they use XP, efficiency will rise and a higher quality level of programming code will be reached.

The new Microsoft .NET framework provides a wide scale of new opportunities. .NET stimulates the use of little components. Methods like test-driven development can seamlessly be used from within XP.

Manager and developers need to shift to a closer integration of both fields. Both are unaware of each others activities and responsibilities.

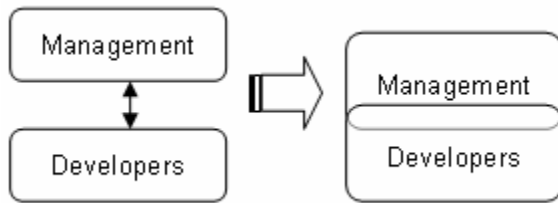


Figure 1: shift to closer integration between management and development

Basket Builders plans to expand in the future. Currently, more work than manpower is available. When a company grows there normally is more need for formal procedures and standard guidelines. This only strengthens the call for a structured testing approach. If the new testing plans will fit in the current process, new developers can immediately use them. Eventually this enables a smoother transition.

## SECTION IV: LITERATURE STUDY

When taking the first step for defining a suitable testing method for Basket Builders it is important to define the focus points for testing.

Based on the issues and requirements described in the previous section we analyzed the possible solutions and theory on testing. Focus was on finding theory that suited the following characteristics of the organization:

- Activities
- Goals (also for the future)
- Magnitude

### Basic principles

After studying the wide and large theoretical part of testing, one can identify some basic principles that can be found throughout the literature. These principles are widely accepted and were evaluated many times. They are guidelines which can always be used when creating a suitable test approach.

1. The cost of fixing an error in the software rises exponentially as the project progresses. As Boehm observed in 1987: "Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase." [9].
2. Testing needs to be integrated throughout the entire software process. Examples that subscribe this are DSDM<sup>4</sup> [29], RUP<sup>5</sup> [30] and the TMap<sup>6</sup> [19] testing method.
3. Code reviews are widely used and respected. Software inspection, which was invented by Mike Fagan in the mid 70's at IBM, has grown to be recognized as one of the most efficient methods of debugging code [8]. It is argued that software inspection can easily provide a ten times gain in the process of debugging software [8].
4. Testing is an important process that is performed to support quality assurance. Testing activities support quality assurance by gathering information about the nature of the software being studied [6,7].

---

<sup>4</sup> Dynamic Systems Development Method [32]

<sup>5</sup> Rational Unified Process [4]

<sup>6</sup> Test Management Approach [18]

5. The process of testing produces many artefacts. Artefacts from the testing include the execution traces of the software execution with test cases. These artefacts can be conserved for future (regression) testing [6].

### The process

Testing is one of the many parts of Software Engineering. It is not possible to see testing without any of the other parts of Software Engineering, in particular the software process.

Because testing needed to be integrated in the current development method it is wise to analyse how well-known methods apply testing.

Small projects tend to be highly iterative both because synchronizing the developers requires less effort and because the management structure allows more direct feedback [5].

The extra effort normally involved when applying formal methods is likely to pay back. However this takes time and patience because the methods need to be learned and controlled first. Also, they typically involve extra effort in the form of additional deliverables.

We started to look at the PRINCE2 methodology because it is used as overall management method. A management method alone is not sufficient; a development method is needed as well. Methods like DSDM, RUP and Extreme Programming were researched, in particular how they deal with testing. Testing needs to be integrated in both the management approach and development method in order to be successful. Management has to plan the testing activities and the tests need to be executed during development. A *dynamic* approach like DSDM and the *control* emphasis of PRINCE2 seem very different at first glance, but when looked at in more detail they have a certain overlap and shared goals. Both handle quality and testing in the overall process. Both suggest that quality is based on pre-determined quality criteria. During testing these criteria will be validated [3].

In cooperation with Basket Builders we explored the different development methods and extracted the key features from these methods.

### *PRINCE2*

PRINCE2 is a project management method that is applicable to many kinds of projects, not just ICT-projects. It specifically deals with changes in the project environment that influence the success of a project [2].

PRINCE2 is clearly an overlapping method. It prescribes a phased approach.

Above all, PRINCE2 is very flexible. It is a set of tools that can be used where and whenever appropriate.

### *DSDM*

DSDM is an iterative development method. It is an extension of Rapid Application Development (RAD) [32]. In DSDM the customer is part of the development team. DSDM also focuses on dealing with a changing environment, especially changing customer requirements. This is facilitated by making timeboxing and prototyping essential parts of the project lifecycle [3].

Workshops in which requirements are defined and tested form a central role within DSDM.

Requirements are prioritized by MoSCoW<sup>7</sup> lists [32].

### *RUP*

RUP is a collection of best practices from the field of software development.

RUP is an iterative process, in which there is a specific focus on software architecture, business modeling and design.

The method prescribes using testing as a continuous workflow during the entire project life cycle [4].

### *Extreme Programming*

Extreme Programming (XP) generally focuses on technical work, whereas the PRINCE2 generally focuses on management issues [11].

The XP approach contains 4 basic project management variables:

- Cost
- Time
- Quality
- Scope (!)

It is very interesting to see that scope is defined as a project variable, for this normally is considered as a constant factor. If the scope is dynamic it seems difficult to manage the project. In XP the project team gets to control one of the 4 basic variables. This is decided by the customer who gets control over the other three [12].

It is a lightweight method in the sense that there are little overhead and additional project assets that need to be produced.

XP focuses on the short future. It is not suitable for bigger projects. We doubt that XP is suitable for middle size projects.

The 4 basic principles of XP are [10, 13]:

- Communication
- Simplicity
- Feedback (i.e. by testing, reviews, etc.)
- Courage

XP prescribes the following solutions based on these principles; these solutions are basically the best software practices that are taken to the extreme [10]:

- Metaphors as means for communication
- Unit Testing – code based testing of small pieces of code
- Design as an iterative process – by refactoring
- Pair programming
- Collective ownership – everyone in the project owns the code

Feedback is vital; and the most basic and critical feedback is that of Extreme Testing [16]. Extreme Testing will be discussed further in this chapter.

---

<sup>7</sup> Must have (o) Should have, Could have (o) Won't have [32]

## Available testing methods and techniques

The following techniques and methods were analyzed during our research:

- The testing phases: Unit testing, Integration testing and Acceptance testing
- Functional testing (type of testing)
- TMap (testing method)
- Exploratory Testing and Extreme Testing

These methods and techniques dictate the literature on testing. There are other testing techniques like Context Driven Testing, Risk Based Testing [27], etc. However, these are basically forms of Exploratory Testing. Furthermore, there are other testing techniques that we did not study because they are not applicable for small projects.

### *Unit testing*

Although early and frequent testing is very important, there is little guidance to date in terms of the specifics of the testing process [14]. In particular, XP requires Unit testing, with a strong emphasis on early and frequent testing during the development process [14].

Unit tests let developers evolve the system rapidly and with confidence and functional tests give customers and developers confidence that the whole product is progressing in the right direction [16].

Unit testing might well be the most agreed upon software best practice of all time [8].

A maintained suite of Unit tests [20]:

- Represents the most practical design possible.
- Provides the best form of documentation for classes.
- Determines when a class is "done".
- Gives a developer confidence in the code.
- Is a basis for refactoring quickly.

Next to Unit testing, there are two other testing phases:

- *Integration testing* [31]
- *Acceptance testing* [24]

### *Functional testing*

Functional testing [25] is a type of testing. One tests the functionality that the application needs to provide.

The difference between Unit tests and Functional tests is that Unit tests tell a developer that the code is *doing things right*, whereas Functional tests tell a developer that the code is *doing the right things* [20].

For Functional tests the specifications are provided by the customer. Functional tests can also be completely done by the customer himself [25].

### *TMap*

TMap is considered as the standard (in the Netherlands, and more and more abroad as well) when it comes to testing. TMap is a formal and above all extensive method that does not have to be applied in total but can be used as a toolbox and guideline. This especially goes for small organizations. In studying TMap, we focused especially on the techniques offered by this method.

Formal methods are needed most for assuring sufficient test coverage when testing multiple application paths that originate from possible property settings (for instance for the activity when testing a User control for the \*Net Toolbox) or possible variable values in the CMS core code.

TMap offers several formal testing techniques that are appropriate for these kinds of tests such as data cycle test, dataflow test and algorithm test [18, 19].

TMap is a testing method executed simultaneously to developing. The test activities typically take 30%-40% of the whole development time. TMap uses a clear test strategy to set the goals about what is important to test. Based on quality attributes and a risk analysis, a well defined approach can be used. The method clearly states the difference between whitebox & blackbox testing [19]. While Unit testing is mainly concerned with code-testing, whitebox testing goes further and even includes evaluating technical designs. This corresponds with the importance that faults are detected as soon as possible.

TMap also deals with the other basic principles. The author of the method states that: "For all types of testing the main activities are planning, preparation and execution" [18]. And just like Boehm observed, TMap also confirms the following statement: "It is known that rework effort on defects increases exponentially per development phase". It is way too extensive to use it completely within a small company like Basket Builders. Even the short timescale makes use of all the aspects impossible. In the method this is also recognized. The following statements are part of the method description:

"For white-box testing it contains too many activities. Only in highly circumstances will all activities be applicable" and

"Choices have to be made, since it is impossible to test a software product completely; 100% coverage on all functionality and quality characteristics is perhaps possible in theory, but no organisation has the time and money to do it" [18, 19].

#### *Exploratory testing*

A more loosely way to test is the testing technique Exploratory testing. Exploratory testing is simultaneous learning, test design, and test execution [17].

*"Exploratory testing is any testing to the extent that the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests."*

*(James Bach) [17]*

The technique is based on the touring bus principle. People on the touring bus take a personal route during a stop but always come back to the main course. So in terms of testing: there is a roadmap that needs to be followed but there must be room to further explore interesting parts that are revealed during the testing activity itself. The case with extensive test scripts is that they sometimes tend to lose significance after they have successfully run once. This is the case with static functionality.

In Exploratory testing a plan is made for each test. In this plan the overall testing strategy (from

the Master Test Plan) is made specific for the part of the system that is the subject of the test: the risks and critical parts are identified and where applicable specific testing techniques are prescribed.

The roadmap for the test is also used by the tester as a guideline for the test report.

#### *Extreme testing*

Extreme testing, the testing approach of XP, focuses on Test Driven Development: this basically means that tests are written before any code is developed.

In XP, ideally, every test should be automated. But it is not always worth automating every test. [16] Fully automating GUI testing, for instance, does not work well most of the time. One ends up spending too much time adjusting the tests to the many small (mostly cosmetic) changes that are made to the use interface [23].

When deciding which tests should be automated, one must always assess the following three criteria [23]:

1. Automating this test and running it once will cost more than simply running it manually once. How much more?
2. An automated test has a finite lifetime, during which it must recoup that additional cost. Is this test likely to die sooner or later? What events are likely to end it?
3. During its lifetime, how likely is this test to find additional bugs (beyond whatever bugs it found the first time it ran)? How does this uncertain benefit balance against the cost of automation?

The currently developed testing framework by Kent Beck, the founder of XP, with his implementation NUnit [35, 21] can be used to test the software.



## Supporting tools

Automating tests is only possible on a very limited scale. In the end, Functional test will have to be manually defined. However, they can be defined in code or scripts. These can then be automatically executed time and time again.

*"Automating chaos leads to automated chaos"*

[www.testforum.nl/viewtopic.php?t=12](http://www.testforum.nl/viewtopic.php?t=12)

Automatic testing is possible for testing the HTML code, Stored Procedure tests, code format (macro) tests and stress/performance tests.

Our criteria for selecting a tool were based on: usability, availability of information and popularity.

### *NUnit* [22]

NUnit is a free Unit testing framework for all Microsoft .Net programming languages. Unit testing can easily be integrated in the development code and executed automatically when necessary. NUnit is widely accepted as the standard framework for Unit testing.

NUnit is relatively easy to use and also provides advanced Unit testing for more skilled users.

The Unit tests can easily be executed by using a small Windows program.

### *VS NUnit* [34]

VS NUnit provides the same functionality as NUnit. A big difference is that the Unit tests are not executed from a separate program but from Microsoft Visual Studio .NET itself.

### *NUnitASP* [35]

NUnitASP is an extension of NUnit specially for testing ASP.NET web pages. NUnitASP focuses on web functionality like buttons, forms and dropdown lists.

One big advantages of NUnitASP is that the testing code does not to be integrated within the development code. The Unit tests directly call the front-end of the web application.

NUnit is free but very limited

### *HarnessIt* [33]

HarnessIt is a more advanced (commercial) tool for Unit testing.. Some highlights:

- Easier integration of testing code.
- Thread testing.
- Difficult to cheat testing.

NUnit should be used first to learn the basics of Unit testing.

### *MS Web Application Stress Tool* [36]

MS Web Application Stress Tool can easily be used to test the performance of websites.

Certain user actions can be recorded and be executed on the tested website. The stress tool can simulate more than 5000 users simultaneously accessing a web application.

The application is free of charge.

### *CSE HTML Validator* [37]

CSE HTML Validator can easily be used to test HTML. The commercial tool tests HTML, XHTML, CSS (style sheets), links, spelling and accessibility.



## SECTION V: RESULTS

### General

The project objective is to set up a testing procedure. During the project we experienced that the overall software process itself is in need of improvement. This process needs to be structured first before a testing approach can be successful. Problems already arise in the phase of delivering specifications to the developers.

A testing approach can not work if the overall software process is not in place [28].

This is an important issue in Test Process Improvement (TPI) as well. It stresses that when improving the test process it is important to maintain a distinction between the test process itself and activities that have impact on the test process [28].

In setting the scope we identified the risk of ending up structuring the entire software process; therefore we needed to set a clear boundary.

We tried to provide recommendations for the software process where this supports the testing approach. The recommendations for the overall software process will stimulate testing efficiency as well.

*"In God We Trust, Everything Else We Test."*

*(Pulsar design philosophy)*  
<http://fozzie.uchicago.edu/~thliu/Pulsar/meetings/031009/14>

In order to promote support and acceptance of the new approach, it was presented to all the employees at Basket Builders.

### Recommendations

The recommendations are divided in 4 categories:

- The overall software process.
- The testing process.
- Specific core based testing.
- Specific web application based testing.

The recommendations have been assimilated into a concrete manual for testing that can be used by everybody at Basket Builders as a roadmap and reference for the testing approach.

A summary of this manual has been added to the Thesis as attachment B.

### Overall software Proces

#### 1. Development method

The testing process is part of an overall software process.

As described earlier, we studied the current project approach and, together with a project manager, we explored several development methods.

Combining the key features of these methods with the requirements at Basket Builders leads to a set of supporting recommendations that promote a structured (testing) approach:

- ✓ The business drivers of the customer organization form the input for prioritizing the requirements. DSDM's MoSCoW method should be applied. Timeboxing should be part of the planning. This enables small releases of features which can be tested independently.
- ✓ Incremental development should be applied (RUP & DSDM).
- ✓ Testing should be integrated throughout the complete project life cycle (RUP & DSDM).

The following principles originate from studying the Extreme Programming method:

- ✓ Testing is an integral part for verification. Especially for a product that is constantly under development (the CMS Core) this is very useful in order to get rapid assurance on the quality of code adjustments.
- ✓ Describe changes and features bases on a metaphor as means for easy communication. This can be applied within the PID.

The points above together should ensure a more integral approach and promote a closer cooperation between the fields of management and development.

## 2. Evaluation sessions

Evaluate every project after it has been completed. The focus within the evaluation sessions should be on parts that went wrong and how to prevent these in future projects? But also, what parts were responsible for the success of a project. It is useful to examine how these can be used in future projects.

During testing, artifacts have been created and these need to be analyzed as well.

### *Testing process*

## 3. Define Master Testplan

A Master Testplan (from the TMap theory) should be included in the Project Initialisation Document. This document includes the following parts:

- Testing mission: Why is testing necessary?
- Prerequisites. Like project delivery date, etc.
- Business drivers
- Risks
  - Business risks
  - Project risks
  - Technical risks
- Quality attributes prioritised
- Testing strategy
- Roles
- Planning testing
  - System testing
  - Integration testing
  - Acceptance testing

The Master Testplan should be drafted by the Project Manager.

For detailed information about the Master Testplan we redirect to TMap [18]. This information is beyond the scope of this Thesis. However we will include a template for the Master Testplan in the testing manual for Basket Builders.

## 4. Dedicated test coordinator

Our advice is to assign one person as test coordinator within a project.

This is very important because this person will be responsible for coordinating the testing activities. He/she will plan out the testing strategy based on the business drivers and (technical) risks.

This task belongs to the head of the project team as this person is in direct contact with the developers and can easily relate to the technical matters of testing. This is necessary because he/she needs to provide the developers with testing techniques and discuss technical content. The testing coordinators can attend courses and transfer knowledge to the other developers.

Tasks of the testing coordinator include:

- Assign the right people to the testing activities.
- Plan tests.
- Analyze testing and take appropriate measures.
- Make the development team aware of the necessity of testing. They must support and implement the overall testing approach.
- Facilitate an evaluation session at the end of a project. The test coordinator must also communicate findings to the other test coordinators within Basket Builders.

## 5. Code reviews

Code reviews should be a fixed part of the development cycle. Code reviews are intended to ensure conformance to standards. They also intend to help disseminate knowledge about the code to the rest of the team. [10]

Furthermore they are intended to ensure that the code is clear, efficient and works. It is a form of early (whitebox) testing. Peer reviews catch about 60 percent of the defects [9].

We recommend to enforce a code review for each integration. These reviews must be included into the project planning because otherwise developers will not have enough time for this. Code reviews should be done by a pair of programmers. It is also valuable to swap between code review partners in order to acquire overall coding insights from multiple perspectives.

## 6. Exploratory testing

Apply principles of Exploratory testing. Each test should be executed according to a specific test plan. This test plan should be based on the content of the Master Test plan combined with the specific (technical) properties of the part that is the subject of the test.

The test plan contains the goals of every test and a risk assessment of the critical features.

This should be the guideline for the test and the results should be reported accordingly. Benefit is that the test is structured and risk based while still leaving enough room for the tester to use creativity and respond to the results of the tests performed without losing track.

A template for the Exploratory test plan is included in the testing manual for Basket Builders.

This approach is compatible with TMap. Because TMap is too extensive for use within small projects of relatively low risk we combined Exploratory testing with some aspects of TMap.

A structured roadmap is needed but tests (test scripts) should never cause higher setup and maintenance costs than cost savings.

Especially within small projects test scripts tend to be too static and of high maintenance. Formal testscripts can always be used within Exploratory tests.

Exploratory tests provide both structure and low maintenance.

## 7. Unit testing

The developer should always provide Unit tests within the code. Not every function needs a Unit test, though. A good guideline for the developers is to write one whenever he feels the need to comment on a function or method.

For ASP.NET development NUnit is a very practical method.

The extension NUnitAsp is an implementation specifically for the use within web applications (this enables the use for Web User Controls).

## 8. Force testing by second person

Next to the Unit tests (whitebox testing), a component should always be blackbox tested as well. The testing approach should enforce that every component is always tested by another

person (other than the developer). The developer should provide the test guideline and risk assessment in the form of the test plan.

When a person tests his own code, it might unconsciously be tested positively. A test performed by a second person can eliminate this problem as well as a possible bias.

*CMS development (the Core)*

## 9. Fixed integration times

Determine standard integration times for the Core code of the CMS. This enforces limited time between integrations so that the source of possible faults and errors can be more easily located.

## 10. Test Driven Development

Apply Test Driven Development (TDD) for the development of the core code for the CMS. This idea originates from the Extreme Programming approach and has been positively evaluated in many projects.

This approach results in more confidence in the correctness of code adaptations and additions. Faults will be much easier to locate.

Maybe it would even be worth the effort to write Unit tests for the existing core code (retrofitting). At the moment it is not possible for us to assess this.

The NUnit framework can be used for TDD.

## 11. Fixed test database

Tests should always be executed using a fixed test database. Multiple sources confirm this theory (for example [26]).

A standard test database is needed for performing the standard set of Unit tests. The values in the database should be aimed at testing the boundary values. For instance, when a functional requirement forces the application to react in a certain way to values higher than 10, the concerning test values should be 10 (do not respond) and 11 (respond).

## 12. Dataflow test

There is need for a formal testing technique for testing the Core. The Core uses a lot of internal properties and variables. This often leads to a

extensive finite set of application paths. A certain degree of test coverage is needed.

From examining the different formal techniques included in the TMap method, we found that Dataflow testing is very suitable for Basket Builders. Dataflow testing is useful when fixating complex schemas for passing variables and properties. Often the extra effort involved in formally describing this (by the developer) is a wise investment. It helps to keep an overview and to ensure a sufficient degree of test coverage.

### 13. Additional recommendation for the Core

During the project we experienced an issue that is not directly related to testing but that we did want to include in our recommendations.

It might be wise to invest in good documentation of the CMS; currently, the documentation of the Core is poor. We experienced that it is very difficult to set up a new installation of the Core and develop a new Control. Because every new employee would have this experience we think it is efficient to improve the documentation.

*Testing of web applications for customers*

### 14. Acceptation site

Organize an acceptance site where the customer can regularly check the intermediate states of the project product so that early feedback is promoted. The customer can be asked to verify certain functionality even though the product is far from finished. For example, an interface test by the customer is often useful.

Early functional testing by the customer prevents expensive product adoptions at the end of the project and minimizes the risks involved following the bias between customer requirements and developers understanding of those requirements.

### 15. Additional tests

Additional tests like performance tests, stress tests (where applicable) should be part of the approach. The additional tests should be planned in the Master Test Plan based on the identified importance of the quality attributes.

### 16. Tool support

The testing approach we recommend is supported by a suite of matching testing tools. This simplifies and enhances the testing activity. The tools recommended in the manual for the testing approach are:

- Microsoft Web Application Stress Tool
- CSE HTML Validator
- VSUnit

### Quality assurance

To assure a certain level of quality (for now and in the future) for the new defined testing approach and its tools, it is essential that new trends, techniques and tools are watched and implemented when required.

We advise to assign one person as testing manager. This person is responsible for the overall testing approach.

He/she has to watch developments from the scientific field of testing and introduce them where useful.

A more practical part of this task is the support of tools. Tools need to be examined and evaluated when appropriate.

Finally, the used testing process needs to be watched and metrics should be acquired and used to evaluate the whole. The testing manager also promotes testing and our recommendations and approach. A manager from Basket Builders can be assigned to the task of test manager.

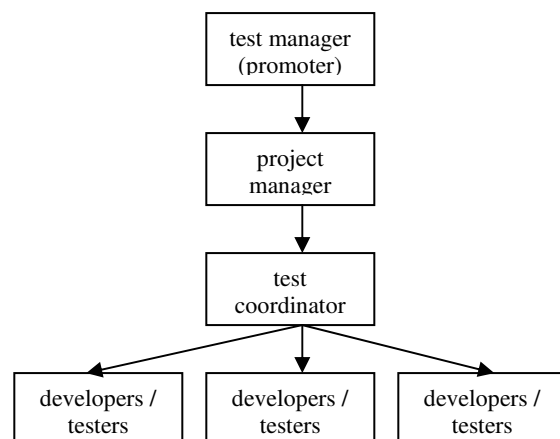


Figure 4: Overview testing roles

## SECTION VI: THE CASE

### Case description

The case involved the development of a control by using Microsoft .NET's framework. The control is a small poll system that can be placed on a website.

This poll control can be added to websites by using Basket Builders' CMS.

In the back-end of the CMS, properties for the control can be set:

- authorisation – for which users will the poll control be visible
- show graph – a graph with results will be displayed

By developing this control we learned about Basket Builders' development approach and the CMS architecture.

From the point of view of testing we focussed on Unit testing. Unit testing is the first step towards semi-automatic testing.

The objective was to assess if the NUnit framework, in particular NUnitASP, is suitable for writing Unit tests.

### Project realization

A functional report was provided. It was not very specific so some additional research was needed. We used NUnit to internally test our code and NUnitASP to externally test on functional level.

The most complicated part was integrating the poll control into the front-end.

### Findings: evaluation

We experienced a steep learning curve but eventually the result was successful.

NUnitASP is a very handy framework, especially for web applications. The test cases are not integrated into the program code so there is a clear separation between program code and test code.

The integration with the CMS was troublesome because documentation was missing.

Software architecture is an important issue when looking at Basket Builders CMS. The whole organization is depending on the CMS because the main activities surround it. Controls need to be integrated in the \*Net Toolbox so a well defined architecture is needed.

The case was too short to give good insight in the testing process. It provided good information about the tools but not about the testing process. Time was not available to execute a project with a longer timescale.

## SECTION VII: CONCLUSION

### *Research questions:*

A well defined testing process will fail if the overall process is not valid. The process has to be well organized before a test approach can be implemented.

Our first research question on the current project approach (Question A: What is the current project approach?), confirmed the statements above. We started by examining the test approach and often we ended up in the software- or management approach. It also became clear that a formal process is not essential in small companies, however structured testing is.

Our second research question (Question B: How are the products currently tested? What procedures are followed?) made clear that Basket Builders applies testing but is not aware of the full potential. Testing is done at the end of a project to find and fix errors; it is seen as a negative activity. In the future testing must be seen as a useful positive means for achieving a higher quality level.

Answering research question C (What available testing techniques and methods are suitable for Basket Builders?) made clear that many testing methods and techniques are too formal and require too much set-up and maintenance effort for use within small, low risk projects.

Question D (Which aspects of the current approach are useful for the testing approach and which should be replaced or rejected?) showed that a Master Testplan can be added to the PID and the task system is a useful way of delegating and monitoring tasks. These tasks can be tested individually.

Question E (How does testing fit into the rest of the software process?) revealed that testing needs to be integrated within all the different levels in the organization, also in the development approach. Management has to plan the activities and developers need to execute them.

Concerning the final research question (Question F: Which tools can be used to support the testing activities?), automating the whole testing approach is not possible. Structured planning and execution of testing is needed. Tools can only support these activities.

The main research question resulted in a testing approach tailored for use within a small web company, more specifically Basket Builders. To support the approach we drafted a manual containing templates, checklists and guidelines for the testing activities.

### *Overall discussion:*

Exploratory testing is more suitable for small projects where fewer risks are involved. More lightweight methods should then be used to save time and money. Web projects are typically low risk. TMap also recognizes Exploratory testing as a useful addition in the new TMap book labeled “TMap Rood” (See attachment C).

Another issue is the need for a more lightweight testing approach that can be used with iterative software methods. Smaller projects, with less risk, need an adjusted testing approach. There is a wide scale of techniques and we combined the ones that were useful.

It was very interesting and reassuring to see that the chosen direction for our findings and recommendations very much corresponded with the subjects that are now part of the new TMap Rood approach.

### **Glossary of terms**

**back-end** = the part of a software system that processes the input from the front-end.

**blackbox testing** = blackbox tests are based on the functional specifications and quality requirements. The system is evaluated in its eventual form.

**the Core** = references to the core code of the Content Management System of Basket Builders: the \*Net Toolbox

**error** = this is more catastrophic. You get an error resulting from an error condition you did not check for.

**failure** = an anticipated problem. When you write tests you check for expected results. If you get a different answer, that is a failure.

**front-end** = the front-end is the part of a software system that deals with the user. The front-end is responsible for collecting input from the user and processing it in such a way that it conforms to a specification that the back-end can use.

**refactoring** = ongoing improvement of the design of existing code.

**variation** = refers to a specific combination of input conditions to yield a specific output condition.

**whitebox testing** = testing based on the program code, the program description or the technical design. It is aimed at the internal operation.

## SECTION VIII: REFERENCES

### Literature references

1. [Basket Builders 2002] *Basket Builders Projecthandleiding 1.0* (Internal Report)., David Smits.
2. [Pink Elephant 2000] *De kleine Prince2*, Mark van Onna, Brigit Hendriks, Günther Schraven.
3. [DSDM 2000] *Using DSDM with PRINCE2*, DSDM Consortium 2000
4. [Rational 2003] *Prince2 and RUP: Loose coupling works best*, Russel Norlund.
5. [IEEE 2000] *A Software Development Process for Small Projects*, Melissa L. Russ and John D. McGregor, IEEE Software September/October 2000.
6. [Harrold 2000] *Testing: A Roadmap*, Mary Jean Harrold, *22nd International Conference of Software Engineering*, June 2000.
7. [Marick 1997] *Classic Testing Mistakes*, Brian Marick, Testing Foundations.
8. [IBM 1999] *Software Testing Best Practices*, Ram Chillarege, Center for Software Engineering, IBM Research, April 4th 1999.
9. [Boehm 2001] *Software Defect Reduction Top 10 List*, Barry Boehm and Victor R. Basili, Software Management, January 2001.
10. [Beck 1999] *Extreme Programming Explained*, Kent Beck, Addison Wesley Publishing Company.
11. [IEEE 2001] *Extreme Programming from a CMM Perspective*, Mark C. Paulk, Software Engineering Institute, November/December 2001, IEEE Software.
12. [IEEE 2001] *Extreme Programming: The Good, the Bad, and the Bottom Line*, Robert L. Glass, November/December 2001, IEEE Software, Loyal Opposition, Computing Trends.
13. *How XP solves Testing and Quality Assurance Problems*, [http://www.xptester.org/\\_ZABLE\[0\]\\_ta b/9/excerpts/xpsolvesqa.htm](http://www.xptester.org/_ZABLE[0]_ta b/9/excerpts/xpsolvesqa.htm).
14. [Parrish/Jones/Dixon] *Extreme Unit Testing: Ordering Test Cases to Maximize Early Testing*, Allen Parrish, Joel Jones and Brandon Dixon.
15. [Ynchausti] *Integrating Unit Testing Into A Software Development Team's Process*, Rany A. Ynchausti.
16. [Jeffries 1999] *Extreme Testing: Why aggressive software development calls for radical testing efforts*, Ronald E. Jeffries, Software Testing & Quality Engineering, March/April 1999, [www.stqemagazine.com](http://www.stqemagazine.com).
17. [Bach 2003] *Exploratory Testing Explained*, James Bach, 2002-2003, [james@satisfice.com](mailto:james@satisfice.com).
18. [Pol/Teunissen/Veenendaal 1997] *A Test Management approach for structured testing*, Erik van Veenendaal en Martin Pol, UTN Publishers 1997.
19. [Pol/Teunissen/Veenendaal 2000] *Testen volgens TMap*, Martin Pol, Ruud Teunissen, Erik van Veenendaal, UTN Publishers 2000.
20. [IBM 2001] *Testing, fun? Really?*, Jeff Canna, IBM developerworks, March 2001, <http://www-106.ibm.com/developerworks/library/j-test.html>.
21. [Beck] *Simple Smalltalk Testing: With Patterns*, Kent Beck, First Class Software Inc.
22. [Beck 2004] *NUnit 2.0*, Kent Beck, 2002-2004, <http://www.nunit.org/index.html>.
23. [Finstarwalder 1999] *Automating Acceptance Tests for GUI Applications in an Extreme Programming Environment*, Malte Finstarwalder.
24. [Millar/Collins] *Acceptance Testing*, Roy W. Millar, Christopher T. Collins.
25. [Holcombe/Bogdanov/Gheorghe] *Functional Test Generation for Extreme Programming*, Mike Holcombe, Kirill Bogdanov, Marian Gheorghe.
26. [MSDN 2003] *Get Test Infected with NUnit: Unit Test Your .NET Data Access Layer*, Steven A. Smith, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnasp/html/aspnet-testwithnunit.asp>.



27. [Rosenberg/Stapko/Gallo] *Risk-based Object Oriented Testing*, Linda H. Rosenberg, Ruth Stapko, Albert Gallo.
28. [Koomen/Pol 2000] *Test Process Improvement*, Tim Koomen, Martin Poll, Praktijkreeks Software testen, 2000.
29. *DSDM Consortium*,  
<http://www.dsdm.org/>.
30. [Kruchten 2004] *The Rational Unified Process: An Introduction*, Philippe Kruchten.
31. [Chan/Chen/Tse 2002] *An Overview of Integration Testing Techniques for Object-Oriented Programs*, W.K. Chan, T.Y. Chen, T.H. Tse.
32. [Stapleton 1997] *DSDM, the method in practice*, Jennifer Stapleton, Addison Wesley Professional.

### Software references

33. HarnassIt, <http://www.unittesting.com/>
34. VSUnit,  
<http://www.relevancellc.com/vsnunit.htm>
35. NunitASP,  
<http://nunitasp.sourceforge.net/>
36. Microsoft Web Application Stress Tool,  
<http://www.microsoft.com/technet/itsolutions/intranet/downloads/webstres.msp>
37. CSE HTML Validator,  
<http://www.htmlvalidator.com>

## POSTFACE

Originally we started with another project for our final assignment. Unfortunately, this proved to be a project that in our opinion could not be justified as having sufficient academic level. Luckily we acted quickly and were able to switch to the eventual assignment. Better agreements on the actual assignment should have been made from the beginning.

After 3 months of hard work we are happy that we were able to conclude this extensive project as scheduled despite the week of work that was lost. Ultimately though, the first week is more an introductory period.

Overall we have to say that we were very pleased with this assignment for our Master Project. It presented us the challenge we wanted and the possibility for an extensive research on an academic level. We studied most of the technological developments in the Software Testing domain. Also the project very much suited the theory already learned during the master Software Engineering.

We learned that a tester needs to be a good communicator. He or she needs to build bridges between the different stakeholders and provide information about the current state of the project. Testing is not only about software anymore but also about people.

Overall the project contributed to being able to see the theory in its right frame and perspective.

A lot of the classical faults were illustrated in practice: testing only at the end of a project, insufficient communication with customer, no frequent deliverables, too little attention for documentation, etc.

Because of the limited time available, planning was essential, even more so because the assignment was a cooperation between 2 students. The planning proved to be effective; in the beginning of the project we drafted a Plan of Approach in which we defined clear timeboxes and goals. This planning was monitored and worked out in more detail every week.

Furthermore the study expanded our knowledge and trained our capability. We believe our knowledge has a strong foundation and now it is time to gain experience and train our capabilities.

Below, Bart Ferweda, project manager at Basket Builders has reflected (in Dutch) on the project and its usability for Basket Builders. We have deliberately not translated his reflection.

*"De stageopdracht van Peter en August is om een testmethodiek te beschrijven voor kleine softwareontwikkeltrajecten. Naar mijn mening zijn ze in hun opzet en uitwerking daarin geslaagd. Door niet alleen een inventarisatie te maken van de huidige testtechnieken, maar ook te kijken naar alternatieve ontwikkelmethoden en projectmanagement technieken, is het testen in een breder perspectief getrokken. Voor Basket Builders heeft dit een drietal belangrijke voordelen opgeleverd.*

*Ten eerste is er een testmethodiek gekomen die voor onze organisatie efficiënt en toepasbaar is. Daarnaast is onze eigen ontwikkelmethode en projectmanagement methodiek onder de loep genomen en verbeterd. Het derde punt is misschien wel het belangrijkste. Door testen een integraal onderdeel te maken van de projectmanagement methode die Basket Builders toepast, is er geen ontkomen meer aan: testen is binnen projecten van Basket Builders een zeer belangrijk en noodzakelijk onderdeel geworden.*

*Voor de toekomst verwacht ik dat de testmethodiek die Peter en August hebben geïntroduceerd zal leiden tot efficiëntere ontwikkeltrajecten. De praktijk zal het nu uitwijzen, ik ben benieuwd."*

*Bart Ferwerda  
Projectmanager  
Basket Builders*

On the following pages, a personal reflection and conclusion from both authors can be found:

"Overall, I am very pleased that we found a challenging Master project that matched our study very well. In such a small period of time we learned a lot and were able to really dive into the subject of testing and several related subjects.

One of the difficulties in writing the Thesis was balancing between two, seemingly contrasting, criteria: On one hand the findings have to be described in a very concise and brief way, but on the other hand everything has to be justified, clarified and motivated. The Thesis could have easily spanned over 200 pages. We tried to capture the essence of the matters described.

We were asked to give an evaluation in the form of grades. My personal opinion is that, from an academic point of view, this is pointless. Without a common reference frame, the individual gradings by the students have little meaning. Everybody handles a different personal motivation, only this motivation is interesting.

**1. Quality of the research results: 8**

We examined and verified an overview of the current situation, issues and requirements. Our recommendations resulted from comparing this to the results from our extensive literature study on testing in small, low risk, projects.

Unfortunately it was impossible to entangle the evaluation of the approach within a Basket Builders project during the graduation assignment. This would have given more clarity on the quality of the results; however this was an unrealistic goal.

**2. Quality of the Thesis: 9**

I am convinced of the quality of the results and the Thesis. Otherwise, I would have made adjustments. Very convenient was that at the end of the project, because of good planning, we had some time left for reviews by experts and fine tuning the Thesis accordingly.

The Thesis is now reviewed by several experts, Basket Builders employees and our mentors. Especially the experts were positive and all the other feedback has been processed. This makes me very confident about the quality of the document.

**3. Difficulty of the research question: 7**

It was challenging to find answers and recommendations that were both useful and reflect an academic vision. However the research question itself was not very difficult, it involved identifying the problems and filtering the available theory for what is suitable for low risk web projects. Extra difficulty was maintaining the scope.

**4. Relevancy of the subjects of the study for executing this project: 8**

A lot of topics from the subjects of our study were illustrated and applicable: Software Architecture, above all Software Process and also of course Software Testing.

For instance the most important message from Software Testing was that the overall software process needs to be well organized before the testing process can be improved. This was very relevant indeed.

Concerning Software Architecture, the importance of architecture as a stable basis was illustrated through \*Net Toolbox, a product that is continuously being further developed.

Looking back at the study I can say that it was a very intensive year in which a lot of theory was handled. During the semesters there was limited time to really dive into a subject; this Master Project did provide this opportunity.

The year has gone by quickly but I now have the feeling that I'm much better prepared and reached a higher professional level. We learned to look at Software Engineering from a broader perspective. I am now more confident to enter the professional business and plan to explore the possibilities of the job market in the field of testing."

(Peter Lamers)

"After finishing the intensive theoretical part of the study, the Master Project looked like a real challenge: The learned theory needed to be applied at Basket Builders. Converting theory to practice was a real challenge. Another challenge was defining and controlling the scope of the Thesis. Testing is a broad subject and overlaps with many other fields of Software Engineering. Filtering unnecessary information was always needed to keep the scope fixed but made the project very intensive. Still we managed to stay focused and the overall project ended successful with a firm combination of testing techniques especially for small, low risk, projects.

The following evaluation was asked:

**1. Quality of the research results: 8**

Controlling the scope and the wide literature made the project a real challenge. The results are from my point of view very interesting because they describe a testing process specially designed for smaller, low risk projects. During meetings with several people from the field we were able to talk to them with the same understanding and abstract level and made me feel certain about the quality of the research results.

Because lack of time it was not really possible to evaluate the new testing process during a real life project. I really wanted to see if the testing process accomplished its business drivers.

**2. Quality of the Thesis: 9**

When looking back at my documents written during previous internships, this time a much higher quality level has been achieved. My other internship reports sometimes stayed too vague. The overall structure is strong and compact which results in straight to the point recommendations. Many people concluded that the Thesis is enjoyable to read while still having a high academic level.

The continuous feedback between Peter and myself made it also possible to achieve a higher quality level. We both share a critical view and all subjects needed to be justified first before it could be added to the Thesis.

**3. Difficulty of the research question: 8**

Answering the research question was quite a challenge. Most testing techniques apply to big projects and are less suitable for small web companies. The conversion from less useful to very useful made the project pretty difficult sometimes. Again the scope was a constant factor that needed to be controlled to maintain focus within the project.

**4. Relevancy of the subjects of the study for executing this project: 7**

Most relevant subjects of Software Engineering are: Software Architecture (this was relevant for Basket Builders' CMS), Software Evolution (Basket Builders' CMS is constantly evolving), Requirements Engineering (a critical subject to make a project more accepted by its users), Software Process (always necessary during software projects) and Software Testing (our main research field). I was able to see and understand the different subjects within Basket Builders. Still I felt that the relationship between the different subjects is out of balance. Some subjects like Software Architecture and Requirements Engineering seemed much more practical than the learned theory of Software Process and Software Testing.

When ending my last graduation I felt that I was missing the overall picture of software development and wanted to enhance my position on the job market. The study Software Engineering gave a good opportunity to look from a broader academic level. Now after finishing this study I feel more confident about the different subjects of Software Engineering and learned a lot that will be interesting for future employees.

Focusing on testing made me realize the good opportunities offered by testing. In the future testing will become a way to differentiate. It changed my view dramatically and hopefully others will follow soon!"

(August de l'Annee de Betrancourt)

Let us conclude with expressing the hope that the testing approach will indeed also bring Basket Builders to a higher level. For sure we will try to follow the development of the company.

## ATTACHMENTS

### ATTACHMENT A: Screenshots of the Poll User Control

These are two screenshots of the Poll User Control we developed in the case study:

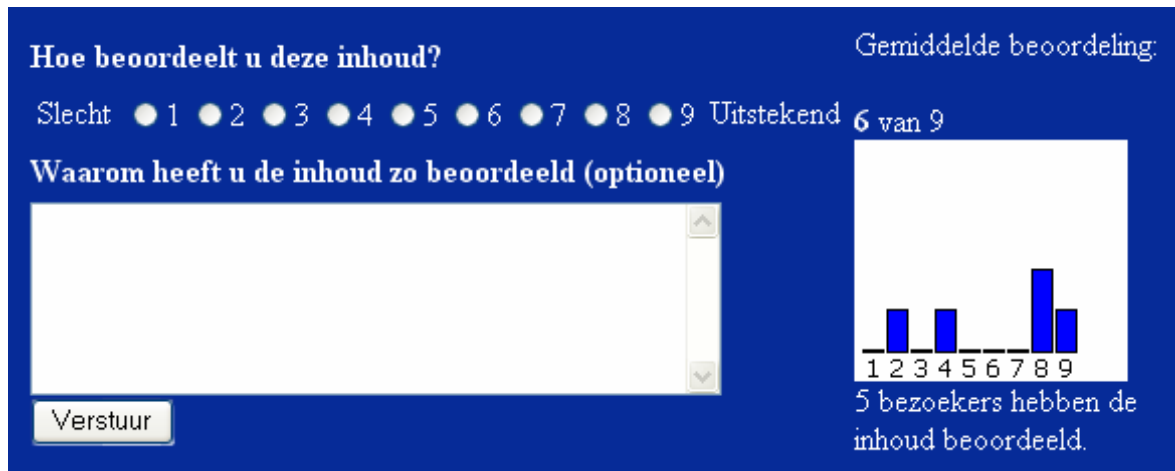


Figure 5: Representation in the front end (the webpage).

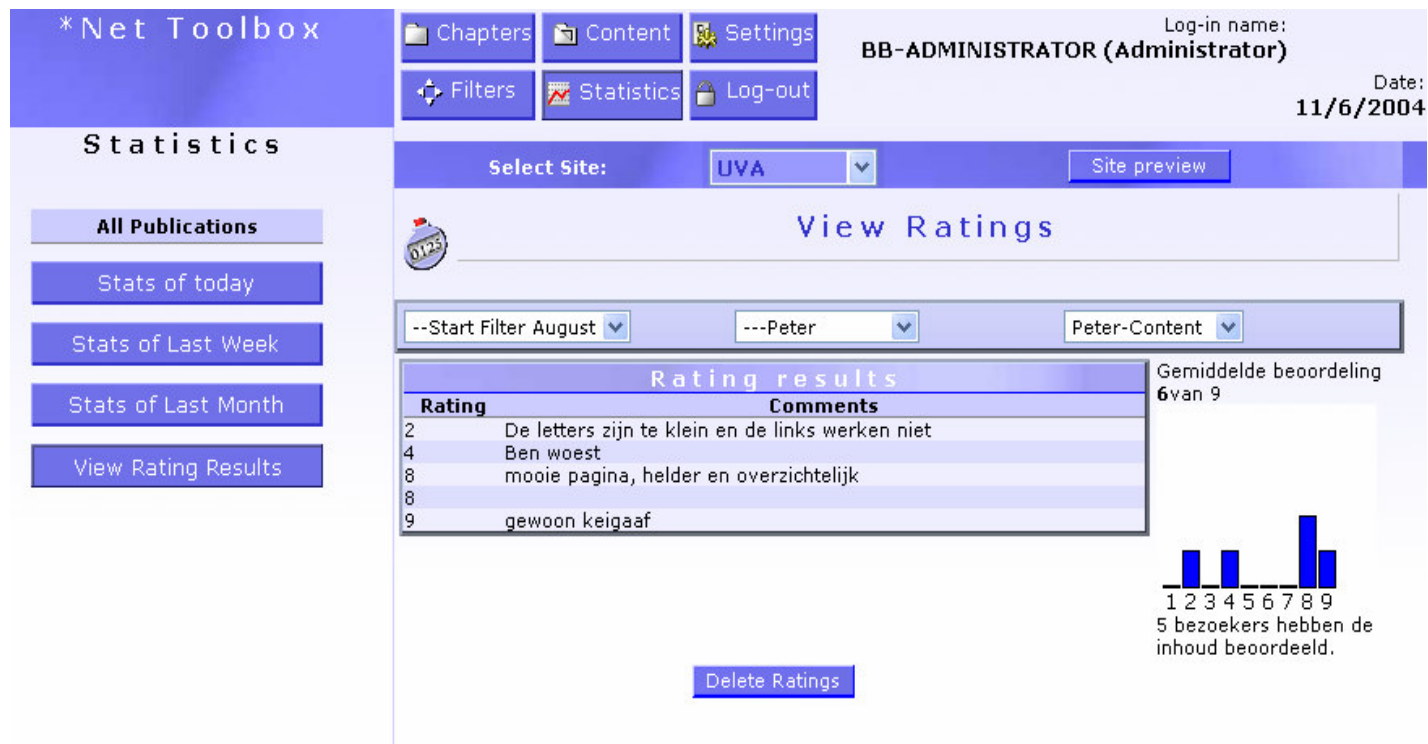


Figure 6: Representation in the back end (the CMS).

## ATTACHMENT B: Summary Testing Manual for Basket Builders

Below, a summary of the manual written for Basket Builders can be found. The manual and the summary are written in Dutch.

---

### Inleiding

De handleiding beschrijft de verschillende onderdelen van het testproces in detail. In de handleiding zijn templates bijgevoegd ter ondersteuning van het proces. Deze zijn niet meegenomen in deze samenvatting.

Het testproces is onderverdeeld in de volgende specifieke onderdelen. De handleiding volgt dezelfde structuur:

- PRINCE2 Projectaanpak.
- Algemene testaanpak.
- Testaanpak voor ontwikkeling van de Core (de \*Net Toolbox).
- Testaanpak voor commerciële projecten.

### PRINCE2 projectaanpak / Project Initialisation Document.

In het PID zal de basis gelegd worden van het testen door middel van het Master Testplan. De volgende onderdelen komen terug in het Master Testplan:

- De business drivers.
- De teststrategie.
- De kwaliteitsattributen.
- De toegewezen resources en planning.
- Rollen en verantwoordelijkheden.
- Mijlpalen.

Het Master Testplan wordt opgesteld door het management.

### Algemene testaanpak

De algemene testaanpak maakt gebruik van het Exploratory testing principe. Het zogeheten Exploratory Testplan is een specifieke invulling van het Master Testplan voor het testen van een bepaald systeemdeel.

Exploratory testing volgt het touring bus principe. De bus (test) volgt een vooraf vastgestelde route zodat men uiteindelijk op de plaatsen komt waar men wil komen. Onderweg is er echter wel de mogelijkheid om even uit te stappen en de omgeving zelf te verkennen.

Mocht meer structuur nodig zijn dan kan men gebruik maken van testscripts.

Het Exploratory Testplan wordt opgesteld door de teamleider.

### Core ontwikkeling

De inhoud van dit hoofdstuk omvat de onderdelen van de testaanpak die speciaal gericht zijn op de (nieuwe) ontwikkeling van de Core (CMS programmacode). De Core ontwikkeling vereist immers een aangepaste aanpak omdat dit een continuerende ontwikkeling van een complex systeem betreft..

#### *Vaste test database*

Testen zouden altijd moeten worden uitgevoerd aan de hand van een vaste testdatabase. De standaard testdatabase is nodig voor het uitvoeren van de standaard set Unit tests en de functionele tests.

### *Test Driven Development*

Test Driven Development is een krachtige manier of effectief foutloze software te programmeren. Er wordt eerst een test geschreven, dan programmacode, om deze test positief te laten draaien. Wanneer alle tests succesvol uitvoerbaar zijn is de code af. NUnit kan gebruikt worden om eenvoudig Unit tests voor Test Driven Development te schrijven.

### *Technieken*

Vanwege het feit dat de \*Net Toolbox functioneert als basis van bijna elk project binnen Basket Builders en de complexiteit van dit systeem is het noodzakelijk om meer formele technieken te gebruiken om de correctheid te garanderen.

Technieken als Dataflow testen (om de gegevensstroom te controleren), Semantisch testen (om de functionele eisen te verifiëren) en Syntactisch testen (om bijv. invoervelden te testen) zijn daar meer van toepassing. TMap kan gebruikt worden als handig naslagwerk om bepaalde technieken te selecteren.

### *Tools*

Het gebruik van Unit testing moet tijdens ontwikkeling zoveel mogelijk gestimuleerd worden. Unit tests kunnen eenvoudig geschreven worden door middel van NUnit en NUnitASP.

### **Overige ontwikkelprojecten**

Naast het ontwikkelen van de Core van de \*Net Toolbox, zijn er natuurlijk de diverse projecten voor klanten (meestal een implementatie van de \*Net Toolbox).


De volgende onderdelen zijn van toepassingen voor commerciële projecten:

- Een acceptatie pagina waar de opdrachtgever de applicatie tijdens het project kan bekijken in huidige vorm. Dit kan gebruikt worden om vroeg delen van het systeem functioneel te testen.
- Unit testing om voornamelijk bedrijfslogica en complexe functionaliteit te testen.
- Code reviews om in korte sessies met een aantal mensen naar de programmacode te kijken en te verantwoorden waarom bepaalde keuzen zijn gemaakt.

## ATTACHMENT C: TMap Rood

Below, an introduction (in Dutch) can be found to the new TMap book called “TMap Rood” which is scheduled to be introduced in the year 2004. This introduction originates from the official TMap website from Sogeti (<http://www.tmap.net>).

### TMap Rood



In het kader van het nieuwe TMap gaat Sogeti de komende tijd vol voor TMap-Rood (werktitel). Dit is een boek dat wordt opgebouwd uit delen die geïnspireerd zijn door diverse nieuwe onderwerpen in de ICT en TMap. De onderwerpen in TMap-Rood zijn:

1. Business driven testmanagement  
Organisaties sturen de ontwikkeling van nieuwe of gewijzigde ICT-systemen op opbrengst, kosten, doorlooptijd en risico's. In het nieuwe TMap wordt het testmanagement nadrukkelijker door deze begrippen geleid, met verbeterde instrumenten voor risicoanalyse, teststrategiebepaling en -begroting en in gedetailleerde testmanagementactiviteiten tijdens de verschillende TMap-fasen.

**TMap implementaties**

2. TMap-light  
Er zijn situaties waarin het voldoende is om slechts een beperkt aantal elementen uit TMap toe te passen. Om in deze situaties het gebruik van TMap te vergemakkelijken is deze lichte variant opgenomen, TMap-light genaamd.
3. Testen en QA van pakkettoepassingen
4. Iteratief/component gebaseerd ontwikkelen
5. Testen van datawarehousing

**Testvormen/technieken**

6. Ketentesten  
Deze vorm van testen wordt ook wel systeemintegratie- of businessintegratietesten genoemd.
7. Exploratory testing

**Methodische relaties**

8. Testen vindt plaats in de context van systeemontwikkeling en -onderhoud. Bij de inrichting van het testen moet rekening worden gehouden met de gekozen aanpak voor de systeemontwikkeling. TMap-Rood voorziet in drie methodische relaties: TMap en DSDM, TMap en Prince2 en TMap en RUP.

De planning is dat het boek midden 2004 uitkomt. Meer informatie: stuur een e-mail naar: [TMap@sogeti.nl](mailto:TMap@sogeti.nl)

Figure 6: Introduction to TMap Rood