Project Planning and Tracking System

A comparative analysis

Arjan C. Schokking

March 23, 2006

Abstract

The Software Engineering Services Department at Philips Research has been using a new software development methodology in its organization for four years. This methodology is called XP@Scrum and is a mix of the Extreme Programming and Scrum agile methodologies. It focuses on incremental and iterative software development coupled with a well-defined estimating and planning framework from the Scrum methodology. It also adds a number of software best practices such as test driven development, pair programming and refactoring from the Extreme Programming methodology. Proper application of these techniques leads to greater flexibility in the software production process and better quality of the developed software as well as higher customer satisfaction.

Philips uses a tool called the Project Planning and Tracking System (PPTS) to plan and track their team based software projects. This tool has been in development for a while and as a consequence has become cluttered with added functionality. This leads to a sub-optimal way of working in many of its primary functions. To address this issue a review of agile methods was done to determine the best methods used to do planning and tracking in other tools.

An analysis of PPTS coupled with the review of a number of its competitors and the information mentioned above leads to a number of suggestions concerning things that should be implemented in PPTS. These have to do with the user interface but also address some issues of lacking functionality when compared to similar tools. Also included, are the features from other tools, which could solve these problems and features, which might otherwise be useful in PPTS. These weaknesses and their fixes are investigated in the masters thesis that will be written after this research report.

A short review of which tools from the comparison are best suited for a given project situation is given too. While this review is not extensive, it does provide a distinction in which tools are best suited for supporting a certain size project.

Preface

This report details the research I did at the start of my masters' thesis project at Philips. I will not say I started this final stretch of my formative years without a little trepidation. However I am glad that my current 40-hour workweek at Philips is not quite as bad as it seemed from the safety of my student perspective even when burdened with an overactive imagination.

But to get back to the matter at hand, Field Marshal Helmut Graf von Moltke is credited with saying: *'No plan survives contact with the enemy.'* and *'Planning is everything, plans are nothing.'* Given this wisdom I can only say that I am extremely grateful to be working on and with a project planning and tracking system. So that now, after 3 months, I am actually finishing this report instead of starting it.

Add to this the excellent help and comments I received from my mentors at both the Technical University of Delft and at the Software Engineering Services department at Philips Research. Thank you for those: Wouter de Jong at the TUDelft, Frank Welberts and Christ Vriens at Philips. This has definitely added to both the quality and readability of this report. Additional thanks go out to my colleagues at Philips not mentioned above, namely: Nicole Belilos, Pascal Vogels and René Barto. Finally thanks go to Joseph Pelrine for allowing the use of his planning circle figures.

So without further ado, enjoy.

Eindhoven, December 2005.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the organization where this research was done, and the origin and purpose of the project.

1.1 Organization

Philips is one of the leading electronics companies in the world and the biggest in the Netherlands. They produce consumer goods as well as a number of products for the professional market such as medical equipment.

Philips Research was founded in 1914 in Eindhoven and has become one of the world's largest private research organizations. Its research is focused on the areas of health care, lifestyle and enabling technologies with two thirds of the research geared toward the Product Design division of Philips and the other third toward more exploratory research.

The System Engineering Services (SES) department at Philips is part of Philips Research Eindhoven. Philips its research division in Eindhoven, often called Natlab, is one of the five research centres that Philips has and is located at the High-Tech Campus. It focuses on developing new technologies by co-operation between various high-tech companies in an environment geared toward technological innovation. SES performs software engineering activities that lead to both prototypes and software. Key value propositions at SES are:

- Knowledge acquisition and anchoring regarding particular Research domains
- Reuse of knowledge/code across group boundaries
- Development of a lightweight iterative and incremental process accompanied with state of the art development environment and tools, Agile programming
- A mix of permanent staff and contractors offering the right capabilities

Within the SES department this project will be done in the Small Work Assignments Team (SWAT). SWAT does small projects mostly for clients within Philips but also from external clients. Both on an individual basis and as small teams.

1.2 Problem statement

SES uses an agile software development methodology. To be more specific they use the XP@Scrum Method, both agile programming, Extreme programming (XP) and Scrum are

discussed in Chapter 2. But in short, both are incremental and iterative development methodologies that focus on small project teams who deliver potentially shippable software at the end of each two to four week iteration.

To support these teams Philips has developed the Project Planning and Tracking System (PPTS)[1]. PPTS is a web based project planning and tracking tool. However a number of other solutions [2] both open source and commercial have entered the market during the last few years. Philips would like to know the position of PPTS compared to these other products. Then use the comparison results as a starting point for creating a new version of PPTS. The new version would remove flaws in the current implementation and provide new functionality. Philips is especially interested in the usability aspects of the project planning and tracking features. Since the user interface is far from optimal. The main methodology that PPTS supports is Scrum and Philips would like PPTS to become the premier tool of developers using this development methodology.

1.3 Assignment

The purpose of this assignment is to assess PPTS and its competitors and analyse their differences and similarities. Then to decide which features might best be integrated into PPTS and determine what can be changed in the current PPTS system to enhance the usability of current features. Special focus with the latter is on the usability of planning features and the user interface. Both of these areas need to be improved. A secondary goal is a comparison of the reviewed packages for general usage by the agile community to provide insight into which tool best suits a specific situation.

1.3.1 Social relevance

The social relevance of this assignment lies in the fact that SES needs the input that the research of the various tools will provide for designing changes to the PPTS tool. At the moment some teams in the SES department are not using the tool any more because of usability issues. This is a situation that SES would like to change. A necessary first step in that process is determining what options for change are available.

1.3.2 Scientific relevance

The scientific relevance of this research lies in the fact that there was no detailed analysis of different agile tooling available at the moment. There have already been a number of requests from various sources for copies of this report when it becomes available. Both by consultants dealing with agile methodologies (see chapter 2) and other people doing research in this area.

1.4 Approach

First a thorough analysis of PPTS and its competitors will be done to find out which features in PPTS are most desirable to implement or to change. To get the most out of this analysis, agile methodologies in general will be studied and a set of test data will be created for use in the comparison of the different tools. Using the test data the different tools will be evaluated and their good and bad points noted. The PPTS tool will also be tested and will serve as

[1] PPTS is discussed in chapter 4
[2] See Appendix A

reference for the other tools. The good points of the other tools combined with the bad points of PPTS can then be used to determine a list of possible changes for PPTS. This will not be part of this research thesis but of the following masters thesis. Very important in the analysis are the usability aspects of the competitors tools compared to those of PPTS. Especially concerning the planning area since it can be improved. For a visual representation of the planned process see Figure 1.1.

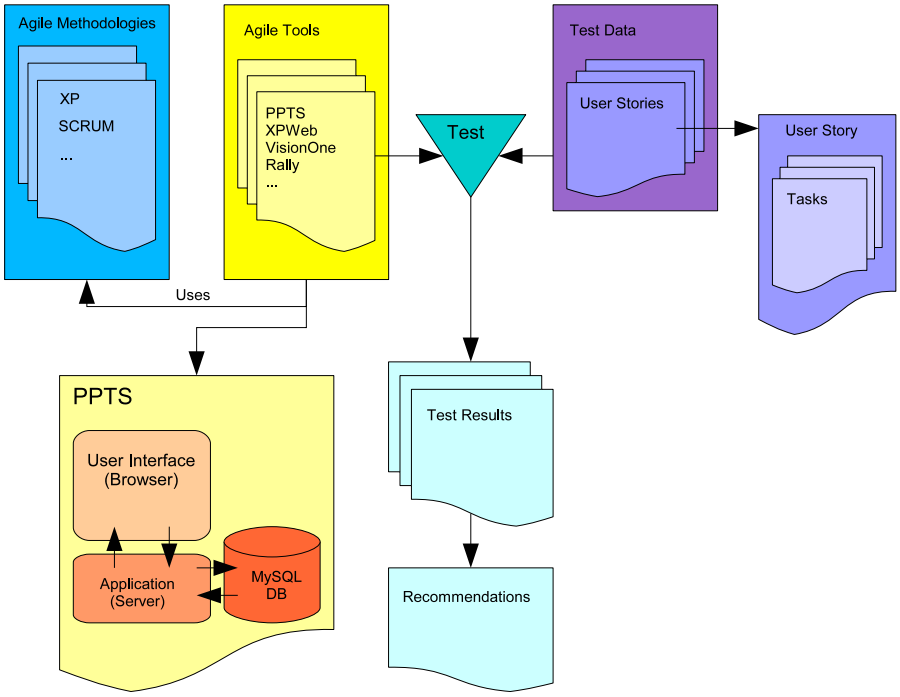


Figure 1.1: The PPTS project work flow

1.5 Planning

PPTS and the software development methodology that is used at SES are based on flexible estimates and progress projections. Since this system is used to plan this project it is hard to say exactly when it will be done. The initial analysis effort is planned for 3 2-week iterations. What will be done after that largely depends on the results of the analysis and how much time is left. See figure 1.2 for the planned time schedule.

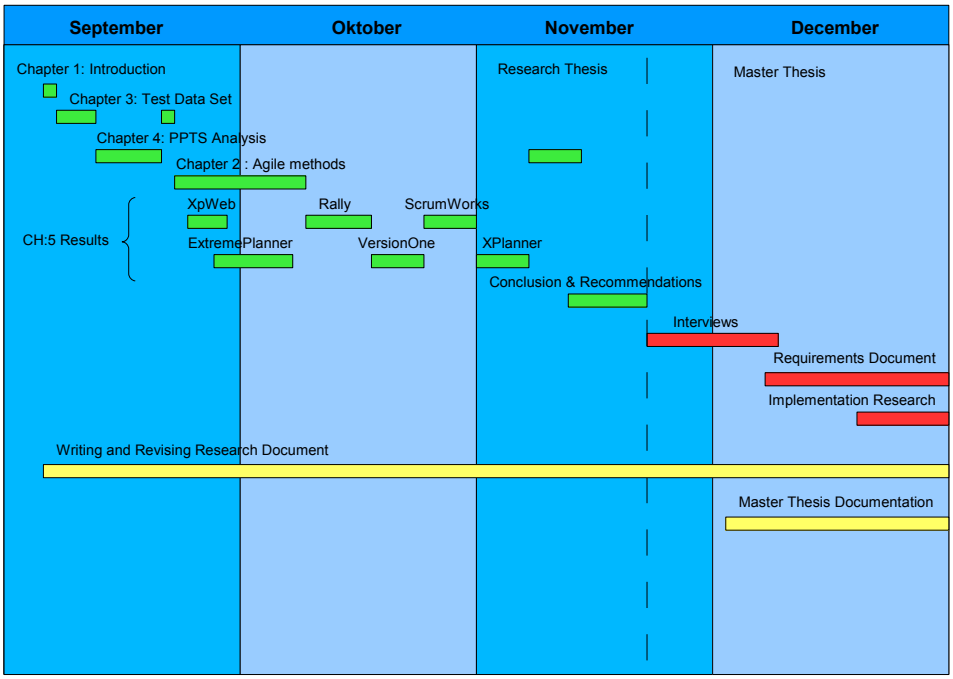


Figure 1.2: Project schedule

1.6 Chapters

The Structure of this document is as follows. This first chapter describes the problem and the approach chosen to analyse it. The second chapter discusses agile methodologies and how these are used at Philips. The third chapter is about the set of data that will be used in the comparison of the various tools. In the fourth chapter the tool currently in use by Philips, the Project Planning and Tracking System, is studied. The fifth chapter discusses the results of the competitive analysis on a tool-by-tool basis. Finally chapter six contains the conclusion.

Chapter 2

Agile Explained

This Chapter explains the history, philosophy and methods that comprise the agile development methodologies. It starts with a review of how agile development came to be, and what the philosophical foundations of agile development are. A few different methodologies are described in section 2.3, followed by a detailed description of the methodologies in use at Philips.

2.1 Agile software development

Agile software development encompasses more then a dozen different agile methodologies. Each has its own niche and is defined by a separate set of key practices. However the agile methodologies also have a lot in common. Where these common factors originated and why agile methodologies are called agile is described in the next sections.

2.1.1 History

Agile software development methodologies have been around for a long time although they were not called agile until recently. An older and more common description of similar processes is iterative and incremental development (IID) [LB03]. One of the first projects where IID was used in combination with software development is NASA's project Mercury in the 1960's. This project resulted in the first United States manned space flight. The software this project used was developed in half-day iterations with a test first programming practice. Test first programming means creating tests for the code before actually writing the code itself. Test first development and working iteratively represent techniques that are incorporated into today's agile methodologies. Members of the Mercury project later seeded the IBM Federal Systems Division (FSD). In the FSD they did a large number of successful IID projects for the Department of Defence proving the approach had value.

It was not until the 1990's that Iterative and Incremental Development began to receive public interest. This attention caused more usage of IID. Which led to the codifying of IID methods into agile development methodologies. A few of the most well known agile methodologies were developed in parallel in the 1990's. Among them Scrum, which refers to a rugby scrum, Extreme programming, XP for short, and the Dynamic System Development Method (DSDM).

Scrum was first applied at the Easel Corporation in 1993 [Sut04] and later developed by Advanced Development Methods (ADM) and VMARK Software (VMARK), both indepen-

dent software vendors [Sch96]. Scrum was formalized during an attempt to explain the lack of breakthrough productivity in object oriented (OO) projects. The software development processes of successful independent software vendors (ISV's) were analysed in an attempt to identify and share high performance practices with customers and other developers.

Extreme programming matured during the time that Kent Beck participated in the Chrysler C3 payroll project [ABB+98]. It was further inspired by the work of Ward Cunningham and Ron Jeffries, both software developers and consultants. It is one of the most extreme forms of agile methodologies. It focuses almost solely on engineering practices as is explained in section 2.2.1. This has gained it fans, and a certain amount of derision by critics who consider it nothing more then formalized hacking.

The Dynamic System Development Method (DSDM) was the result of an effort by 16 developers working with Rapid Application Development (RAD). They were trying to define a standard iterative process to support RAD at a RAD users conference. Inspired by James Martin's work on RAD and its time-boxing features the results of the conference led to DSDM.

In addition to these agile methodologies there are a dozen-plus others, each with its own speciality. Agile software development is not a solution to all software development problems. But it has advantages when used in the right circumstances. Most of the agile methodologies focus on an environment which features rapidly changing requirements because they support the ability to adapt to change as needed.

2.1.2 Philosophy

What is agile software development? As stated before it deals with incremental and iterative development and includes a number of other practices, which will be detailed later. The philosophy behind agile software development is stated in the agile manifesto [CF01]. The manifesto was created during a meeting of 17 top software developers representing a myriad of different agile methodologies. The text of the manifesto, which is short enough to quote, follows:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> - ***Individuals and interactions*** *over processes and tools*
> - ***Working software*** *over comprehensive documentation*
> - ***Customer collaboration*** *over contract negotiation*
> - ***Responding to change*** *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

Most of the methodologies that fall under the header agile were first defined as lightweight or low overhead processes. Dissatisfaction with those names was one of the reasons that prompted the informal conference, which led to the manifesto. At this moment processes that follow the principles stated in the agile manifesto are denoted as agile, although there is a lot of discussion over whether some methodologies belong there or not. A selection of established agile methodologies is described in section 2.3.

What can be deduced from the agile manifesto is a shift from rigid thinking to flexibility and interaction. It has also been said that following the manifesto moves power back from management to developers. This should not come as a surprise since the authors of the manifesto were mainly developers themselves. Whether this also leads to a larger number of successful projects as opposed to projects using standard software development methodologies,

such as the waterfall model, remains the question. Little scientific data is available on agile project successes when compared to projects using other methods. Some studies are available about agile practices like pair programming [WU01] but they do not touch upon using the entire methodologies.

2.2 Agile concepts explained

Before the methodologies are discussed a number of agile terms that are used in the following sections are explained.

A ***user story*** is like a very small use case. It describes a certain piece of functionality in the system. However a User Story is supplied by the customer and is written down in a few sentences at most. User stories denote features of varying sizes, which are part of the planned application.

Tasks describe actions that need to be taken to implement the functionality described in a user story. This means that user stories are split up in a number of tasks that will provide the functionality it describes. Tasks are specified by the development team and usually consist of coding and related activities.

Ideal days and ***ideal hours*** are the amount of time that a developer works without any interruptions or distractions whatsoever. As such it is an abstract time measure and is often the unit that is used to estimate the size of stories and tasks. A second notion to abstract the size of some piece of functionality from the time it will take to implement is story points.

Story points are an arbitrary unit for estimating size for a thing relative to something else. As such 2 story points denote twice as much work as one story point. The benefit of this notion is that it does not have the reference to time that is inherent in the term ideal day and ideal hour. Additionally there is a translation from these ideal units to real time. This factor is called the Load Factor.

The ***Load factor*** is the multiplication constant that converts Ideal days and hours to normal workdays and hours. Where real hours equal the number of Ideal hours times the Load Factor. Common values for the Load Factor are 1.3 to 1.6, this means that between thirty and sixty percent of work hours is devoted to other things then tasks directly related to the project. For instance: meetings, answering mail and coffee breaks.

An equivalent term to load factor for use with story points and the Scrum methodology is ***velocity***, this term indicates how many story points of work were finished in an iteration or some other amount of time. the term velocity is sometimes denoted in a percentage instead of an absolute number, In this case it usually refers to the difference between ideal time and real time. A velocity of 70% would indicate that 70% of the available time is ideal time, in other words, time spent solely on project tasks. The other 30% is considered overhead.

The smallest pieces of code that can be tested and used in isolation are referred to as units. Usually these are a function or procedure. A ***unit test*** is the test that provides input to a unit and checks the output against the expected values that the unit should return.

To explain refactoring martin Fowler is quoted from his book “Refactoring, improving the design of existing code”

> ***Refactoring*** involves making changes to the internal structure of software so it’s easier to understand and cheaper to modify without changing its observable behaviour

Refactoring is used to keep code easy to understand which helps in extending it later on in a project.

2.3 Agile Methodologies

As stated in the last section there are a number of different agile methodologies, each of which incorporates a different set of best practices from the software industry. The factors, which bind these methodologies together, have been discussed in sections. A list of these methodologies follows as well as a brief description. For more information see [ASRW02].

- Extreme Programming
  Extreme Programming [ABB+98] [Bec00] is explained in section 2.2.1
- Crystal
  The Crystal family[Coc04] contains a number of methodologies. Each is geared toward a different kind of project based on importance and size. These methodologies are commonly called Crystal followed by a colour code, which can be: Clear, Yellow, Orange and Red. Where Clear is for small projects and the Red for big projects. At this moment only the Crystal Clear and Crystal Yellow methodologies are well defined and have been used in practice. Both of these have a number of common policies. They are:
  - Incremental delivery
  - Progress tracking by milestones based on software deliveries
  - Direct user involvement
  - Automated regression testing of delivered functionality
  - Two user reviews per release
  - Workshops for product and method tuning at the beginning and middle of each increment
- Scrum
  Scrum[SB02] is discussed in section 2.2.2
- Adaptive Software Development
  Adaptive Software Development [Hig00] is aimed at the problems encountered when running large and complex software development efforts. It, like all other methodologies, advocates iterative development but with a strong prototyping aspect. It aims to provide a framework, which will provide stability in a project without negative effects on creativity and emergence.
- Feature Driven Development
  Feature Driven Development [PF02] covers only the design and building phases of development. It can however incorporate the other activities in a software development effort. It features iterative development with a number of best practices, which have been effective in the industry. This methodology emphasizes quality, process monitoring and frequent, well-defined deliverables.
- Dynamic System Development Method
  DSDM [Sta97] is a framework for the Rapid Application Deployment (RAD) software development methodology. It focuses on working within fixed time and resource constraints, and varying the amount of functionality accordingly. DSDM defines five phases. These are the feasibility study, business study, functional model iteration, design and build iteration and implementation phase. The last three phases are iterative.

- Additionally there is a methodology called Agile modelling which features an agile approach to performing modelling activities and Pragmatic programming which is a set of programming best practices published in 'The pragmatic Programmed' by Andrew Hunt and David Thomas.

The next sections will discuss the agile methodologies that are used at Philips on a number of projects including this one. Since a more in-depth understanding of these methodologies is needed in the rest of this paper their descriptions will be more thorough.

2.4 Extreme Programming (XP)

Kent Beck and Ward Cunningham developed Extreme programming during a number of projects they worked on together. It is based on a number of software best practices that they identified as working well in the projects they did. They bundled these practices and put them to use a number of times, for example in the Chrysler C3 experiment [ABB+98]. The practices were formalized into 4 key values and 12 key practices which comprise the Extreme Programming methodology. Also each member of a project team usually is assigned one of a number of roles defined in XP. These roles will be discussed before the values and practices.

2.4.1 Extreme programming roles

Each member in an XP team can have one or more roles, these are defined as follows:

- Tracker :
  The tracker keeps track of the progress of the developers and takes action if things start to go wrong. This may include setting up a meeting about the problem, contacting the customer or asking someone else for help. It is like a stripped down Scrum Master role, see the description of Scrum roles in section 2.2.2
- The customer :
  The customer writes user stories and specifies functional tests. The customer also decides what will be implemented at the iteration planning and provides more information about specifics if needed.
- The Programmer :
  The programmer writes tasks, estimates both stories and tasks, writes the unit tests then implements the user stories.
- The coach :
  The coach's task is primarily to watch everything and make sure that the project adheres to the XP discipline.
- The Tester :
  The tester is responsible for running the functional tests, processing the results and taking action on those results.
- Doomsayer :
  Points out when the sky is falling down and the team is probably about to die or the project to fail, also known as the critic. This is not an entirely official role but such a person crops up in almost all teams and can be a valuable indicator of trouble spots in a project.

- The Manager :
  The manager schedules meetings, makes sure they proceed according to plan, writes the minutes and does other management related tasks. He does however not tell the team what to do since it is self-managing.

Not all of these roles are present in every project, nor will they always be done by separate people. Usually, however, most of them will be represented in a project team in one form or another.

2.4.2 Extreme Programming values and practices

The four values of XP are communication, feedback, simplicity and courage. The 12 key practices embody these key values and are discussed below with a description of each since they are used in a number of SES projects as a part of their XP@Scrum methodology.

Kent Beck published a book called 'Extreme Programming Explained' [Bec00] in 2000, which detailed these key practices and how they could be implemented in software development. Following is a brief discussion of these practices and what they entail.

1. The Planning Game:
   This practice originated with another agile methodology namely Scrum but is used in a number of agile methodologies. Scrums version is however slightly more detailed since Scrum focuses more on planning.

   The planning game is played on two levels, the release and the iteration level. It involves a close interaction between the customer and the developers. In the release planning game the customer starts by defining a number of user stories that have to be in the release. These are usually written down on cards. When the customer has specified all the user stories that he can think of at that moment the developers estimate the size of each user story. The customer then prioritises the user stories according to business value and he decides what scope the release will have. This estimation coupled with the load factor that the team uses leads to an initial time estimation on the first release. This will be a rough estimate. But after a few iterations the team will have a better understanding of how many stories they can complete during an iteration and as a result the estimates will become more accurate.

   On the iteration level the customer and developer choose a number of stories from the release to work on in the next iteration. The developers then break each story up into the tasks and estimate how many ideal hours each task will take. Then the team checks how many ideal hours there are planned in the iteration and multiply this by the load factor to see if the team has enough resources available to finish the stories. If there are not enough resources or too many, stories get added or removed until the iteration is full.

2. Small Releases:
   XP works with small releases. Typically one release every 2 months. This makes it easier to steer and control a project, especially in conjunction with the planning game. It also allows for more frequent and better feedback about a functioning system even if it does not incorporate all of the functionality yet.

3. Metaphor:
   XP uses a metaphor for the project. This is a word or phrase that describes the project

and is shared by the customer and developers. It should be a concept, which both parties can relate to. An example would be 'Desktop' or 'Peer-to-Peer' to describe either operating system functionality or file sharing between two equal entities.

4. Simple design:
   XP emphasizes designing the simplest solution that still fulfils the functional requirements. Unnecessary complexity, extended functionality and gold plating are removed as soon as they are found. This keeps the system simple, flexible and easy to re-factor. All of this makes it easy to add new code to the system.

5. Testing:
   XP uses test driven development. This means that before any production code is made there should be unit tests available which test all normal situations and any boundary conditions. Once a working test set is created the actual production code is written. Unit tests are used to verify the production codes correctness and the code is updated if any bugs are discovered.

6. Refactoring:
   XP works best with clean, simple and easy to understand code. To attain this goal it requires refactoring of any code which does not meet these standards whenever possible.

7. Pair programming:
   Pair programming means that two people are working on the same computer. One will be typing, that user is typically called the 'driver' and the other will be watching for mistakes and what needs to be done where, that user is called the 'navigator'. While two people working on the same computer may seem a waste of manpower, research [WU01] has shown that the code produced is typically almost bug free and of a much higher quality then code created by a single programmer. The time saved on debugging and other such activities mean that in total, practised pair programmers are very time efficient. Pair programming also leads to dissemination of knowledge minimizing the risk of project delays when people leave the project and take critical know-how with them.

8. Collective code ownership:
   This means that the code is collectively owned, which means that anyone can change any piece of code at any time. This can lead to concurrency issues and it requires a good code management system to implement.

9. Continuous integration:
   Any new code is integrated into the code base as soon as it is ready. This means that there will usually be multiple builds each day. With new functionality being regression tested and added to the code base on a regular basis.

10. No overtime:
    XP prescribes a 40-hour workweek, in principle no overtime is allowed. If overtime is necessary two weeks in a row it is noted as a bad project smell indicating a problem. The 40-hour workweek is used because after 5 or 6 hours people start to make more errors and because overwork is not viewed as a productive or fun activity, thus lowering the team's morale.

11. On-site customer:
    In an optimal situation a representative from the customer is on the development team to guide the development and answer any questions that they have. This representative is also responsible for creating user stories and accepting any changes that need to be

made during development. If an on-site customer is not available a proxy is the next best solution. This practice is considered a vital part of the XP method even if it rarely happens because of time constraints on the customer. Not having a customer in the team longer communication lines. This leads to slower development and a product that will diverge more from what the customer really wants.

12. Coding standards:
    Coding guidelines should exist and be used consequently throughout the development of the product. These guidelines help keep the code understandable and easy to read.

Having explained the basic XP practices it must be stated that these are theoretical and that no documented XP project has implemented all the above practices. Usually a number of the above practices are implemented that fit in the current development methodology or ideology of an organization. There is little statistical data available on the number of projects that have succeeded or failed using this approach. Although the first indications regarding the use of XP seem positive.

2.4.3 Extreme Programming life cycle

The XP life cycle is divided into five phases. They are called the exploration, planning, iteration to release, finalizing and the maintenance and death phase. A visual representation of the life cycle is given in figure 2.1.

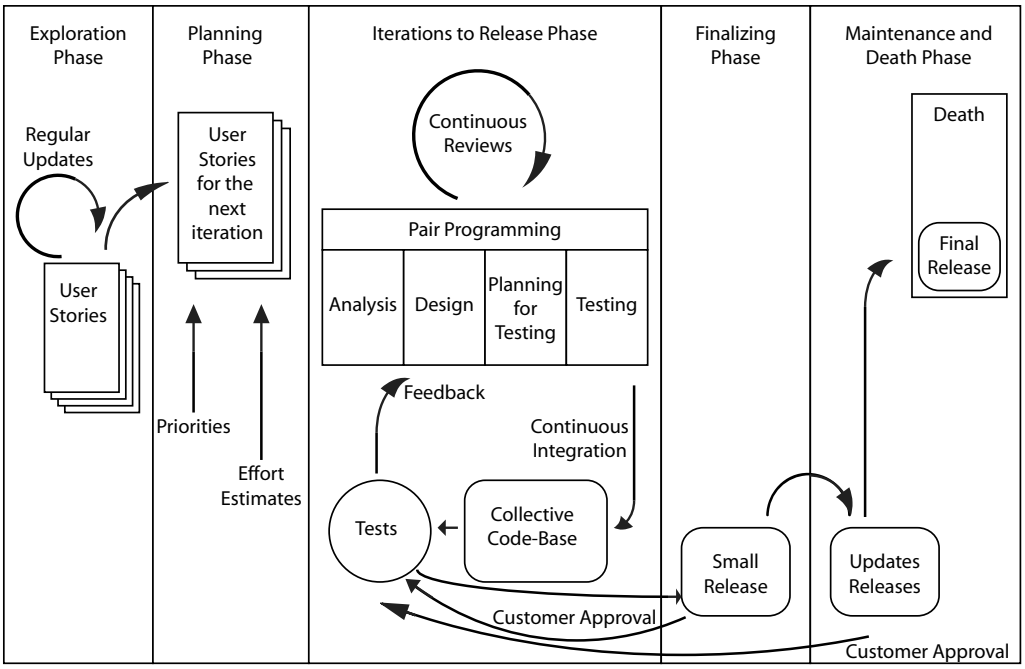


Figure 2.1: The Extreme Programming life cycle

In the exploration phase the customer writes down the user stories that they want to include in the first release. Meanwhile the developers research the technologies needed to implement the project. They also test the architecture and technology by building a small prototype of the system.

In the planning phase the user stories are estimated and it is decided what stories will be in the initial release of the system.

In the iteration phase iterations are run until the first release of the system is ready or in other words until all the user stories are completed and accepted by the customer. The scope of the release can be varied during this phase as long as it is understood that this impacts the release date.

In the finalizing phase the product is readied for release to the customer. Issues that remain unfinished are documented and any last problems are resolved before the system is released.

After release to the customer the project enters the maintenance phase. It consists of continued production of a new release and providing customer support for the current release. It is presumed that dedicated people will be used for customer support since using the development team for this task can seriously affect the speed of further development.

The death phase starts when there are no more user stories left to implement and the final system is made ready for release to the customer. Additional tasks like writing documentation and other support tasks are usually done at this time. Death can also happen if the project is cancelled due to an unsatisfied customer or through a lack of budget or other resource needed to continue development.

2.5 Scrum

Since Scrum is one of the main methodologies that PPTS supports, if not the main methodology, it receives additional attention. The information below is mostly from "Controlled Chaos software development" and "Controlled Chaos: Living on the Edge" by ADM both reports [Met] [Sch96] can be studied for further details.

As stated in the history section, Scrum emerged from an attempt to identify why object oriented (OO) projects where not showing good results in many cases. This research was done by independent software developers most notably Advanced Development Methods (ADM) and VMARK software. These companies believed that writing good software is an art guided by rules of thumb, tips and techniques. In essence that writing software is a creative process. They began to research what made their software development more productive then that of comparable vendors who used Capability Maturity Model (CMM) certified processes. They asked scientist at the DuPont Chemical Advanced Research Facility, chemical process specialists, to review their system development process. These specialists had a large body of knowledge about processes and process automation. They concluded that software development contained many processes that were undefined and unrepeatable instead of well-defined, repeatable and predictable processes[Bro87].

Software development efforts had, in their eyes, more in common with empirical processes then defined ones. A defined process is one that has no unknown elements and which can be repeated an unlimited number of times yielding the same results each time. Therefore a defined process can be fully automated. An empirical process is a black box. With no knowledge of the inner working of the process only its inputs and outputs can be observed. From those inputs and outputs it is sometimes possible to derive a model. Empirical processes, like those

in the chemical industry, are tightly controlled keeping input values between certain bounds to make sure nothing goes wrong.

The final recommendation of the scientists was that a number of tight measurement and control processes be defined to keep track of the software projects parameters. ADM and VMARK software then studied the software development processes in a few highly chaotic development environments, for example at Microsoft [Cus98]. This study led to a number of requirements with which a software development effort would have to comply. A system must:

- Be tolerant of chaos and be able to respond in a flexible manner to expected unpredictable events
- Be constantly measured and refocused with controls
- Assume the product is continually evolving, or in other words non-static, as the knowledge, comprehension and thus the requirements of the owners and users evolve
- Maximize communication and information sharing.
- Create the best possible software while dealing with requirements, quality, timetable and resource constraints

Scrum establishes controls over software projects to insure that these requirements are met. The controls a Scrum project uses to achieve this are the following:

- The Backlog:
  A list of all requirements, which should be fulfilled in the completed product based on the customer's current knowledge. The items on a backlog are user stories
- Objects and Components:
  Self-contained reusable things
- Packets:
  A group of objects within which a story will be implemented. Coherence between objects in a packet is high but coherence between individual packets is low
- Problems:
  Things that need to be resolved by the team to implement a story within one or multiple objects, this can include bugs
- Issues:
  Concerns that must be resolved before a story is assigned to a packet or before a problem can be solved by changing a packet
- Solutions:
  Answer to a problem or an issue
- Changes:
  Activities that lead to a problem or an issue being resolved
- Risks:
  Risks associated with a story, problem or issue.

2.5.1 Scrum development phases

Of all the controls the Backlog and the risks associated with stories are the most crucial. How the Scrum process measures and tracks these controls will be explained next. The Scrum process is divided into three phases:

The first phase is called the pre-game or planning phase. It contains 2 sub-phases: Planning and architecture.

In the planning sub-phase the user stories, which describe the functionality of the system, are created by the customer and estimated by the development team. These estimates and the backlog of stories are continually updated throughout the project as additional information becomes available. The team members are selected and other resources like tools are acquired. Risk assessments, control issues and training are also part of the planning phase.

In the architecture sub-phase the high level design of the system is planned based on the user stories that are currently on the product backlog. This activity can be done in a sprint, the term for an iteration in scrum, or be done outside the process. It is not depicted as a separate phase in Figure 2.2. Plans for the initial content of the first release are also made in this sub-phase.

The development phase is usually the longest phase in a project. This phase is treated like the black box in an empirical process with its controls, inputs and outputs. Development is done in sprints, which is the Scrum term for an iteration. At the end of each sprint the product is incremented with the produced functionality. The sprint is then reviewed in the sprint review. System design and architecture evolve during sprints as does the product backlog. The customer can always add or remove stories from the backlog. Consecutive sprints are run until the product owner is happy with the functionality in the product. The final phase, which is often called the post-game or implementation phase, is then started.

In the post-game phase the product is prepared for release. Anything that still needs to be done, like final integrations and documentation ,is done. When the product is ready it is released to the customer. Preparing a release, like the architecture definition, is usually done in a separate sprint and as such also is not depicted separately in figure 2.2. A picture of the Scrum process is shown figure 2.2.

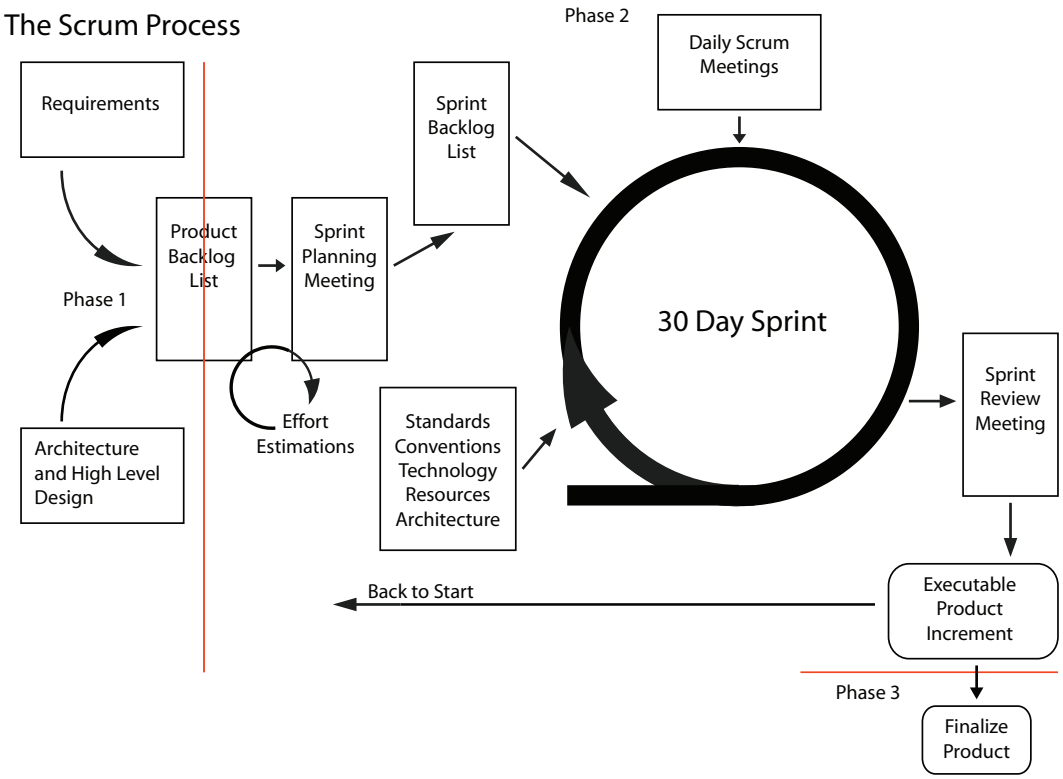


Figure 2.2: The Scrum process

2.5.2 Scrum Roles

Scrum has three well-defined roles and three key stakeholders. The three roles are: The Scrum master, product owner and Scrum team. The key stakeholders are: The customer, the users and management. The roles in Scrum are better defined then the ones in XP because Scrum focuses on management and control where XP focuses on engineering practices. This focus is why a person will usually have a single role and no roles are left out in a typical project. If they were it is likely that the project will run without the necessary controls to keep it heading toward the proper goal

- The Scrum master:
  The Scrum master is responsible for the application of the Scrum methodology in the project. The Scrum master confers with the team, customer and management. He is responsible for removing obstacles so the team can work at peak efficiency. The Scrum master is also responsible for keeping track of projects progress.

- The product owner:
  The product owner is officially responsible for the project. He is in charge of the product backlog and is appointed by the Scrum master together with the customer and management.

- The Scrum team:
  The Scrum team is the project team responsible for achieving the goals of each sprint. The team is self-organized and helps with planning, estimation and the creation of the functionality.

- The users:
  Users are people who are going to use the system under development. Their commitment to the project in collaboration with the customer is important to guarantee that implemented features meet the users needs.

- The customer:
  The customer represents the person or company who ordered the product. the customer is involved in creating the Products Backlog.

- Management:
  Management makes the final decisions about the project and defines what standards it will use. Management tracks progress and help select the product owner among other things.

2.5.3 Scrum Practices

Scrum practices refer to management practices. The main practices in Scrum are:

- The product backlog:
  This is the backlog for the entire product, or as much of it as is known at a certain time. It consists of a list of user stories that have been defined by the customer.

- Effort estimating:
  The estimation of initial effort required to implement each user story in the backlog. This is also done on a task level at the start of a sprint. Where the tasks in each of the stories in the sprint are defined and estimated.

- The sprint:
  A sprint is the Scrum term for an iteration. The management tools used in a sprint are: the sprint planning meeting, the sprint backlog and the daily Scrum meeting.

  - The sprint-planning meeting:
    This is a two-phase meeting. In the first phase all participants of the project decide the goal and scope of the sprint. In the second phase the development team plans the product increment decided upon in phase one.
  - The sprint backlog:
    A backlog with user stories that have to be completed in the sprint.
  - The daily Scrum meeting:
    A stand-up meeting at the start of each day in which each member of the team answers the following questions: What did you finish yesterday?, what problems did you encounter? and what are you planning to do today? This allows the team to keep track of what's going. The meeting also helps detect and solve problems before they can delay development.

- The sprint review meeting:
  The results of the sprint are demonstrated in this meeting. Decisions about the next sprint are made, and any changes to the product backlog are discussed and implemented.

These practices are the core of the Scrum methodology. They can help manage and control chaotic software development processes and they provide excellent information about the status of a project. The development practices of XP when combined with the management processes of Scrum are called XP@Scrum. XP@Scrum is a popular way of working with agile methods. How these two methodologies overlap is discussed in the next section.

2.5.4 Future Development

The development of scrum is not standing still and while the cutting edge of scrum, which deals with overlapping iterations and other measures to increase efficiency, is beyond the scope of this report. Readers can view the last presentation by Jeff Sutherland on Scrum [Sut05] for more information.

2.6 XP@Scrum

As explained in the previous sections XP emphasizes programming practices. XP emerged from the coding patterns movement and it addresses issues with code quality and readability. On the other hand Scrum emerged, at least originally, from Japanese product development processes and focuses on managing the delivery cycle. Luckily both methods complement each other very well. Extreme programming practices can address the lack of methods for dealing with low-level issues in Scrum. Scrum management practices can in turn be used to solve high level planning issues in XP. The only overlap between the two methodologies is the planning game, which was directly taken from Scrum by XP to give it a rudimentary estimation and tracking ability. This overlap causes no problems since its implementation is the same in both methodologies. All the other practices the two methodologies have are not present in the other. All Scrum and XP practices are shown in Figure 2.3. The red circle describes XP with its practices listed to the right. The blue circle depicts Scrum with its practices to the left. The purple section is the overlap and as described above only holds the planning game.

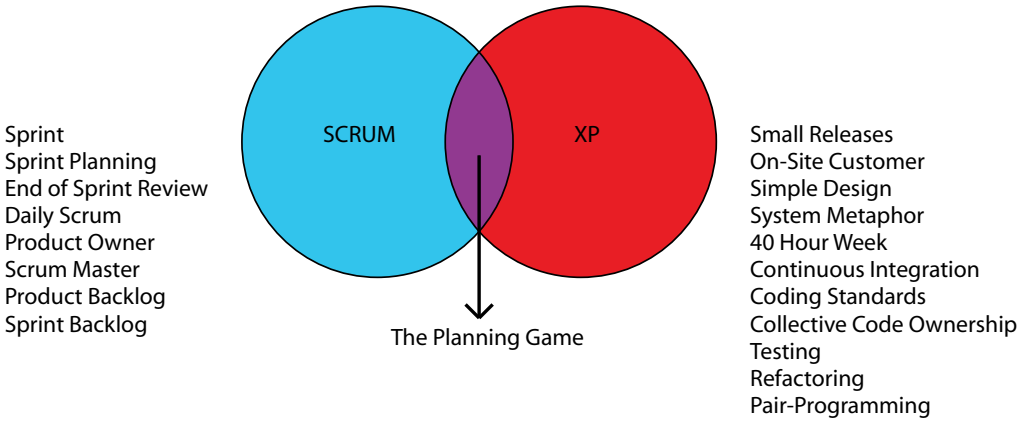


Figure 2.3: Overlap of XP and Scrum practices

The Software Engineering Services department of Philips uses the XP@Scrum approach to facilitate their software development efforts. As with most real world implementations they do not follow XP to the letter, but use those pieces that best fit the organization. Planning and tracking in SES is however entirely based on Scrum. Although with a number of extra procedures, which are needed to retain the Capability Maturity Model level 2 certification. There are some features in PPTS that deal with resource planning within Philips and are not in XP or Scrum. These will be highlighted in chapter 3. How Scrum and XP work together is detailed in the next section.

2.6.1 XP circles

As seen in figure 2.4 the XP practices can be arranged in a number of concentric circles [1] representing different domains within which the XP practices fall. This is a useful representation when thinking about how Scrum and XP work together. The four circles, from inner to outer, are called the coding, team, and process and product circle. Each denotes the area within which the practices in that circle apply. The coding circle comprises the testing, refactoring and pair programming best practices. The team circle contains continuous integration, coding standards and the collective ownership of the code base. The process circle with the 40-hour workweek, simple design and the system metaphor practices. Finally the last circle, called the Product circle, contains the planning game, on-site customers and small incremental releases

[1] Joseph Pelrine, Copyright 2003, MetaProg GMBH

practices.

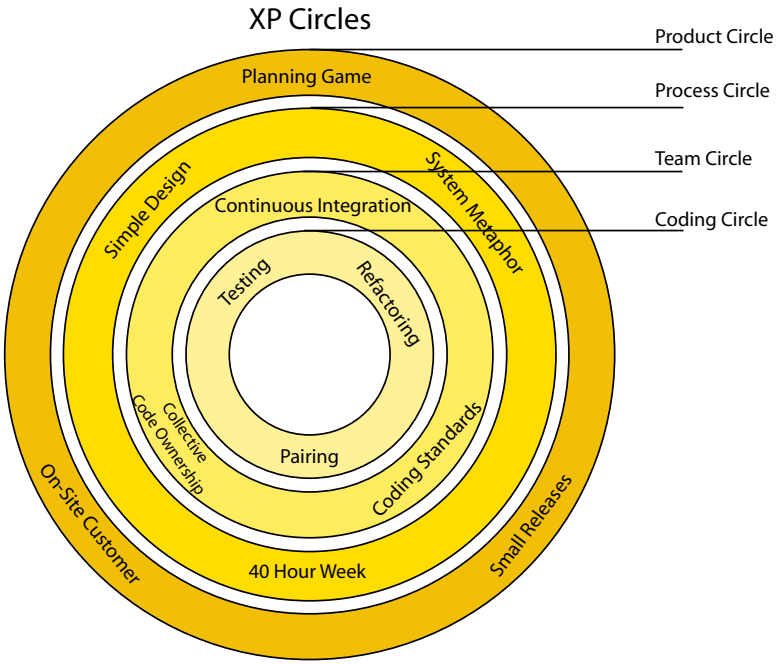


Figure 2.4: The XP Circles

A Scrum project runs a number of sprints, which incrementally deliver software. It also has a daily cycle for producing that software. Now as stated in chapter 2, XP@Scrum are XP practices wrapped with a Scrum project planning and estimating approach. In the next section this practice will be visualized and explained for the Scrum life cycle figure.

2.6.2 XP@Scrum

Scrum and XP do not really overlap except when it comes to the planning game as was explained at the start of this section. This fact allows for an easy representation of how Scrum wraps the XP practices. A number of the XP practices are wrapped by Scrum practices, which have to do with the daily work that is done each day of an iteration. These practices are depicted in figure 2.5 as the smaller circles at the top of the figure. The inner circles depict the XP coding and team circles from figure 2.4. The outer blue circle stands for the daily Scrum. On the other hand the XP practices in the larger circles have more to do with practices regarding the iteration are wrapped by the Scrum practices which encompass planning and tracking for a sprint. This is shown by depicting the product and process circles from figure 2.4 inside the bigger blue circle at the bottom of figure 2.5, which represents a Scrum sprint. This way the XP engineering practices are combined with Scrum planning and tracking, the former represented as the inner circles the latter as the encompassing blue circles.

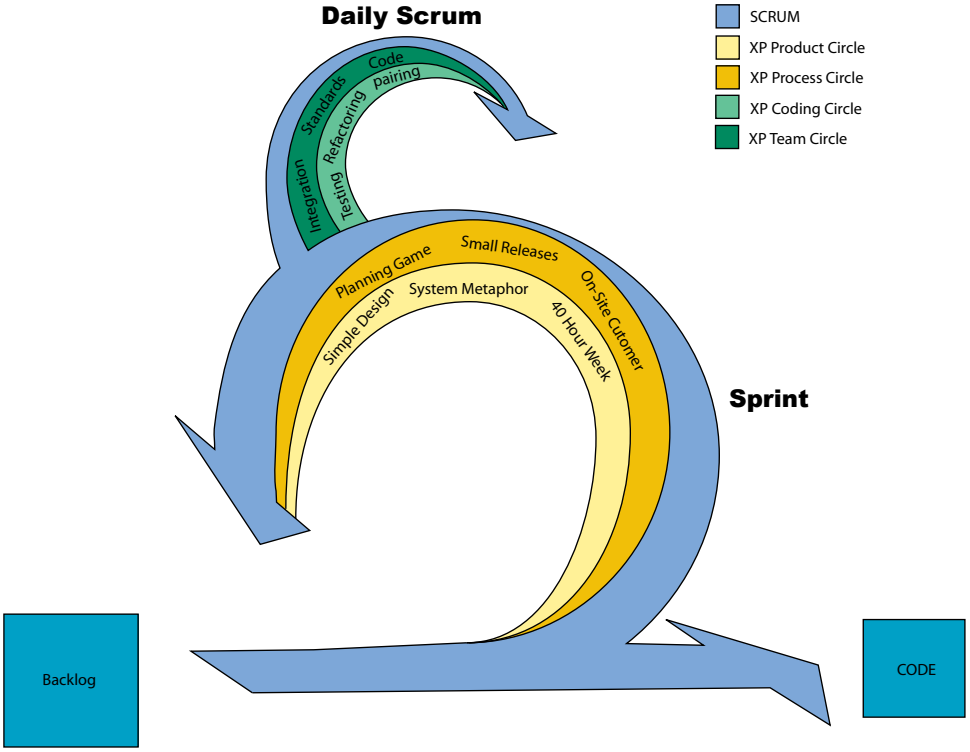


Figure 2.5: XP practices wrapped by Scrum planning and tracking

Chapter 3

Test Data

This chapter discusses the test data that is used in the analysis of the various tools. The basis of the test data and the evaluation criteria that were used are also explained.

3.1 Introduction

To make the comparison, a set of data in the form of user stories[1], their associated tasks, and estimates is needed. This data is needed because user stories and tasks form the basis of projects using the Scrum and Extreme programming (XP) agile methodologies. Stories and tasks are the main things that are estimated and tracked in agile projects. They are also the main objects that all the tools that were analysed use to define and track projects. That is why the data set has user stories and tasks as its basic units. Time keeping and estimations are done in hours.

Since this test data is used to analyse the functionality that each tool offers. It should be as similar to data from an actual project as possible. This allows for an accurate comparison then would occur when using fictional test data. With that in mind, the test data was based on four iterations from the project which initially created the PPTS tool. To start with, only two iterations of the project are used. If more data is needed for proper analysis the latter 2 iterations can be used as well.

The set of user stories that comprise the test data has been edited for clarity. Two items, which did not directly relate to the project, were removed. This concerns user stories for documentation of an internship and a user story used to keep track of hours marked as overhead. Since neither of these provided any functionality to PPTS they were removed. Note, as was explained in chapter 2, that all user stories should describe features specified by the customer. In reality however user stories are sometimes created to keep track of additional overhead tasks in a project. These are the ones that were removed to create a data set, which is as close to an ideal situation as possible.

3.2 Structure

In this section a brief overview of the basic hierarchy of work items in an agile software project is given for the XP@Scrum methodology in use at Philips. This is a generally accepted hierarchical representation of items for this methodology and applies to most other methodologies as

[1]The terms user story and task are explained in chapter 2.

well. The basic project layout is dictated more by how software development works in the real world then by the methods used to develop it and thus is pretty universal. The hierarchical representation of work described below is often called the Work Breakdown Structure (WBS) and functions as the main input of the tools.

3.2.1 Project

A project features a description of the system that is being built and acts as the top-level container for other work items. It is the most top-level description available in tools and usually only has a title and a description as properties. A project generally comprises a number of releases.

3.2.2 Release

By release a software release is meant. This is a fully functional piece of software, which implements a certain of amount of the functionality that is desired by the customer for a project. Releases are usually the highest level at witch planning and tracking for a project occurs. Releases consist of a number of iterations, which incrementally add to the code base of a release, ultimately resulting in a running system. The average time between two releases in an agile environment is between three and six months. The functionality in a release is specified as a number of user stories. This list of user stories is called the release backlog.

3.2.3 Iteration

Iterations were discussed in Chapter 2. Each iteration implements a number of stories from the release backlog. Iterations are the second level on which planning and tracking is done for a project. Iterations last between one and 4 weeks, usually the duration depends on the amount of time between two releases. The length of an iteration should be balanced between giving the development team the time to develop a part of the functionality, and allowing the customer to make changes in between iterations. Iterations continue until the release backlog has been implemented or the owner decides enough has been implemented for the current release. The customer is free to add and remove user stories from the release backlog at any time. But the customer is not allowed to change the user stories in an iteration during that iteration.

3.2.4 User stories

User stories denote pieces if functionality that the customer want to have implemented. They are estimated in ideal hours or story points. Both units are abstract notions that denote relative size. In other words a user story with an estimate of 2 is twice as hard as a user story with an estimate of 1 and a story with an estimate of 4 is twice as hard again. While these units should be an abstract notion of size, in reality this is not that easy. People tend to think about hours of effort they spend on something and not in abstract measures.

3.2.5 Tasks

For each use story in the data set there are a number of tasks which implement the functionality that the user story describes. These tasks all have estimates in ideal hours. The addition of these separate estimates for tasks leads to a more precise estimate for each user story. When all tasks in a story are completed, that story is marked as completed. Tasks usually have an

entry for the how many hours were spent on them and an entry for how many hours the task is still expected to take. The latter measure is referred to as the number of hours to go or, in shorthand, Togo.

3.2.6 Users

There are a number of users, performing different roles, assigned to each project. Some tools allow the user to create teams. However, usually this is done implicitly by adding a number of users to some project. Users have roles in projects. These are defined along the lines of the roles discussed in Chapter 2.

3.3 Assumptions

For testing purposes the assumption is made that 2 people will be working on the project. Both of them will spend 40 hours per week on the project. They are both developers and have the same set of skills plus comparable expertise. The iteration length is set at two weeks. There is one spokesperson for the customer in this project who is always reachable for the development team. To summarize:

- Two people will be working on the project, where each has 40 hours a week to spend on it
- The iteration length will be 2 weeks
- The Test will run over 2 iterations initially, subject to change if it is apparent that more or less iterations are needed to demonstrate the functionality of the tool.

Estimation methods from Mike Cohn new book "Agile Estimation and Planning" [Coh05] will be used if applicable. All other planning and estimation practices will also be used as described in this book.

3.4 Spreadsheet Explained

The complete list of the test data can be found in appendix B. This data contains four iterations. Each iteration has about a dozen stories and two dozen tasks. Usually, while testing, only one project and one release will be created since a release is the highest level of planning available in all the tools. In the following section the contents of the spreadsheet will be explained via a few short examples taken from the spreadsheet. The terms used here were explained in chapter 2.

Work Breakdown Structure: wk21/22	(hours)	(hours)	(hours)	(hours)	
User Story Description	Estimation	Total Initial Effort	Total Used	Total Last ToGo	CA*
1. Upload user photographs	2	2	2	0	Y
2. Graphical overview of user absents incl. print possibility via PDF file (2)	48	16	16	0	N
3. Quality Assurance wk 21/22			2	0	Y
4. Configuration Management wk 21/22			0	0	Y
5. Indirect hours wk 21/22			3.5	0	Y

Table 3.1: Sprint backlog list

The top line in the figure 3.1 shows the iteration and the units of measurement, in this case ideal hours. Following that there are a number of user stories, the ones that have the colour dark orange are not included in the test set because of the reasons stated in section 3.2.4. The estimation header indicates the initial estimate made for a story. The total initial effort is the estimate gained by summing the estimates for each task belonging to a story. Total used hours denotes the hours actually spent on the story and finally the ToGo describes the number of hours still needed to complete the story. Since this is historical data this number will always be zero. The final column is called CA, which stands for customer acceptance. This means that the customer has seen and approved the work product, in this case a finished user story.

2. Graphical overview of user absents incl. print possibility via PDF file (2)				
Create graph		8	14	0
Include holiday balance table on PDF file (on right side of legend)		6	1.5	0
Remove second filter for holiday balance (use first filter instead)		2	0.5	0

Table 3.2: Story tasks list

This second set of entries in the spreadsheet lists each story with its tasks, where the numbers, from left to right, are the initial estimate of each task and the amount of time actually spent on it. Finally, totals are included for both the story estimates and the task estimates. These estimates tend to be different since the estimate on a task basis is more accurate then those based on user stories. This follows from the fact that tasks are smaller then user stories and thus, in general, easier to estimate accurately. When an estimate derived from tasks is available it is used instead of the estimate based on stories.

3.5 Exceptions

There are a number of tools, most notably Rally and VersionOne, which are hosted solutions usable only with a guest account that provides limited functionality. In these cases the given projects and stories will be used to evaluate the tools. Both Rally and VersionOne use test projects based on past real world projects or projects that are comparable enough that it does not make any difference. Readers should be aware however that in these cases other data then the defined test data was used in the comparison of these two tools. If this impacts the analysis it will be noted in the results section.

3.6 Evaluation Criteria

Finally the criteria for the analysis need to be defined. In other words what aspects of the different tools will be looked at and evaluated. First the general aspects are discussed and later on the more specific issues to do with usability, planning and tracking. Any tool which claims to support agile development methodologies will have a certain amount of features related to the different aspects of agile software development projects. Reiterated from chapter 2, those main features are:

- Projects
- Releases
- Iterations
- Stories

- Tasks
- User Management

Each of these features is related to certain pieces of functionality. They are create actions, delete actions and similar items, which are fundamental to the use of any tool. In addition different tools may have functionality which is unique, and therefore, interesting.

There are also a number of abstract aspects related to planning and tracking, which are very important because planning and tracking are the activities which the tools need to support. Planning is done on the level of releases and iterations. Release planning is the normal activity of planning releases by adding a number of iteration to them and then planning the iterations. The planning of iterations is done by adding user stories to them until the amount of effort they require to implement equals the resources available in an iteration.

The somewhat more abstract aspects that will be evaluated are:

- Planning
  - Implementation
  - Metrics
- Tracking
  - Release and Iteration based.
  - Metrics
- Usability
  - Installation of the tool
  - help features / documentation
  - responsiveness
  - layout / UI

Most important to Philips is the usability and implementation of planning and tracking features in the current system. To see where these usability aspects can be improved the basic tasks that are common to all agile projects need to be evaluated. The basic tasks described here are from Mike Cohn's book on agile planning. These are aspects that will be highlighted in the analysis according to how they are accomplished in the different tools. A last aspect, but not an unimportant one, are features that cannot be placed in any of these categories but do add value to the tool. These features will be discussed separately in the results section of the tool they belong to.

After analysing all the tools, and comparing each tool to PPTS, a determination must be made whether each tool:

1. has features that are not in PPTS, and if so, if those features are useful
2. has features, which provide the same basic functionality but are better implemented
3. has some features, which are badly implemented or useless and should be avoided

It will be interesting to see how closely each tool follows a pure agile methodology or whether they have been changed significantly to accommodate practical usage issues or quality bounds like a CMM certification.

Chapter 4

Reference Tool, PPTS

The system in use at Philips is called the Project Planning and Tracking System (PPTS). It will be discussed in this chapter.

4.1 Introduction

The System Engineering Services (SES) department at Philips Research has developed the Project Planning and Tracking System (PPTS). PPTS is used to plan and track software development projects. It has a lot of different features. Some features support the agile methodology at its core, while others stem from the fact that SES is CMM level 2 certified. Others add functionality specific to Philips. The latter two types of features necessitate a number of changes in the tool from one devoted solely to agile methodologies. These changes will be discussed in the analysis.

4.2 Analysis

This thesis is being planned with the PPTS system. That data will be used in the evaluation of PPTS. The main reason behind this is that PPTS does the burn-down graph calculations and its timekeeping on the server side, which makes it hard to simulate a running project. Thus the data from the thesis project itself will be used since it has a similar set-up to the test case. Which means a small project with, in this case, one researcher and the same boundary conditions.

This review of PPTS and the features it offers will be along the lines of the reviews of the other tools in chapter 5, although the analysis will be more extensive. A number of general remarks will be discussed followed by some extra features that do not pertain to planning and tracking. Next, features about planning will be discussed followed by the tools project tracking features. These sections will be subdivided according to the hierarchy of an agile project. Sub-optimal implementation of features is the main point of attention since this signifies trouble spots. These coupled with the well-implemented features from the tool comparison will lead to a number of recommendations for change at the end of this thesis.

4.2.1 General Remarks

It should be noted that, while PPTS is an open source tool, its primary purpose is to support projects within Philips research. Thus it has a large number of features that are specific to this

purpose, such as budgeting, room reservations, expertise and course overviews, and a number of other management related items. These are less useful for external use of PPTS. Since the focus of this research is on planning and tracking section of the tool. These extra features will not be evaluated unless they touch upon the planning and tracking. A number of these features are actually helpful when doing planning and tracking, these will be discussed in the next section.

The installation of PPTS is pretty straightforward since it is a dynamic website. As long as there is an internet server and a MySQL database, nothing should go wrong if the instructions in the installation guide are followed. Initialising the database might be a bit difficult for users without any MySQL knowledge and it could be automated.

4.2.2 Extra Features

PPTS has a lot of extra features that are used in Philips and some of those add value to or are used by the planning and tracking sections of the tool. However there are also features, which do not directly relate to either planning or tracking. These features will not be discussed at length, but they are listed below so the reader has an idea of what is included in the tool.

- Unit Tests : Allows the user to run the unit tests for PPTS to see if any additions broke the code
- Quality Assurance Checklist : Allows the user define checklists for different kinds of Q&A. This feature is needed for CMM Level 2 certification
- Rooms : Allows the user to add rooms that can be reserved for meetings and reserve the room at a specific date and time for a certain period
- Detailed resource allocation : Shows for how much time each employee is booked on which project
- Detailed project overview : Shows details on all projects currently running
- Competences overview : Shows the competences of all employees
- Q&A overview for all projects : shows Q&A issues for all current projects
- Full Time Equivalent of all projects : Shows how many FTE employees are assigned to a project
- Course cost overview : Shows how much each employee has spent on training courses
- Agencies : A list of all agencies that Philips Research works with and or hires employees from, this includes contact data
- Risk Overview : Risk overview for a project
- Budget Overview : Budget overview of a project
- Project Status Report overview : List of project status reports made for a project

Next to these items there are a few extra features, which do interact with planning or tracking aspects of the tool. These will be discussed in more detail next.

- Absents :
  The absents page lets the user indicate days when they will not be at work, due to sickness, holidays or any other reasons. This is a good way to keep track of who will be available for projects, excluding really unexpected events. The system also keeps track of the time that a user is absent during each iteration. Then adjusts the time available to that user to complete tasks in the iteration. This means that PPTS calculates the actual time each user has to spend on tasks that leads to a more accurate planning. It also leads to better tracking during the project since it is easy to look up why someone is not at work. This does of course assume that users will diligently fill in their absences.

- Workload :
  This page shows the workload for people on a project by comparing the amount of work assigned to them to their realizable effort based on historical velocity data of previous iterations. This can be a helpful tool during an iteration to see if someone is shouldering too much, or not enough, work. Discrepancies here are usually the result of inexperience with estimating how much time a certain task will take.

- Resource Allocation :
  The resource allocation graph shows how much of each employees time is dedicated to projects. this allows users to see, at a glance, which people still have hours available, which can be assigned to other projects. Coupled with the overview of competences, discussed next, this gives the project leader a very powerful tool to plan new projects. The project leader can see who will become available when and who is available now and for how many hours.

- Competences :
  This page shows the different competences each employee has in a matrix view. It includes knowledge the user has about: Programming languages, operating systems, problem domains and methodologies they know. This data, in combination with how many unassigned hours employees have left, is very useful when checking if the right mix of talent is available in a project team.

Most of the things listed above relate to initial project planning and project resource management. This is the first stage of planning a project. These features can also be very helpful in signalling changes in resources such as the amount available. Unfortunately this signalling function is not currently clearly integrated into the PPTS user interface.

4.2.3 Planning

This section will highlight the planning features of the current PPTS implementation. This analysis is structured by the manner in which the planning process proceeds. Starting at the creation of a project, creation of the backlog and finally creating an iteration with backlog items and tasks. Another important item in addition to the functionality in the program is the usability and feedback that the program gives a user. This will be discussed too.

Project

Defining a project, and assigning people to work on it, starts a project. These are the base requirements of doing anything in PPTS. An administrator can create a new project on the projects page. The user who creates a project can then fill in all necessary details. A PPTS project allows the user to fill in a lot of customer and date related information. The rest of the

information is mostly concerned with project contacts, dates and descriptions of the project, exact information about the fields can be found in the product comparison matrix in appendix C.

When a project has been created users can be assigned to it. This happens on the users page where each user is displayed with a button labelled 'assignments', which can be used to view and alter the projects a user is assigned to. When assigning users to projects information is needed about who has unassigned hours remaining. This information can be gained from the resource allocation page. This is a separate page, which shows an overview of how many hours each user still has available for allocation to projects. When the administrator has selected the users to work on the new project he then goes back to the users page. There users can be assigned by clicking their assignments button which opens another page that allows and administrator to add project assignments to that person and enter the number of hours they are going to spend on the project.

This process is a bit roundabout, especially since the user needs 3 different pages to actually find and assign users to a project. This action will be relatively rare since new projects are not started that often inside a department and it will be done by the department head who has most of the information he needs memorized. However neither of these things makes the implementation less cumbersome. They just make its impact less.

Another way to accomplish this task, and one that is more logical, would be to have an assign users button on the project page. This would open a window which shows people that are less then 100% assigned to other projects or are about to become available, and how much time they have available. The user can then select them by drag and drop, select box, or whatever other method has been implemented. After that the administrator can fill in the number of hours the selected user will spend on the project and save the changes. Some solution along these lines would certainly make the project planning process easier. But since it is not used much it has a low priority.

There are a number of pages that provide information about projects in PPTS. The first is the main projects page. This shows the selected project, some basic information about it like, start date, end date, the customer and all the users assigned to the project. This page also allows the user to create a new project or modify the selected one. The second project related page in PPTS is an overview page. It provides an overview of all projects, current, planned or previous ones. It also contains some basic information that is limited to people assigned, description and dates. Finally there is a page to which project status reports can be submitted, these are just filled in templates that give the status of the project at a given time. The overviews are limited to the first two pages, this leads to the question why there is no page, which gives a status for each project since there is enough data available in the database, and it is a common feature in other tools. The fact that the data is in the database is deduced from the information available in other parts of the PPTS tool. This information is certainly interesting for project tracking if not especially planning. Therefore, it will be discussed further in the tracking section of this chapter.

Backlog

The Backlog page in PPTS is where the user stories for a project are added, see Figure 4.1. User stories can also be added directly to and iteration but for planning purposes we focus on the Backlog page. Each user story added shows up with an estimate and name. They can be added to any defined iteration and the user can set the priority of stories by switching stories in the list. The position of a story on the list denotes its priority, where closer to the top

indicates a higher priority.

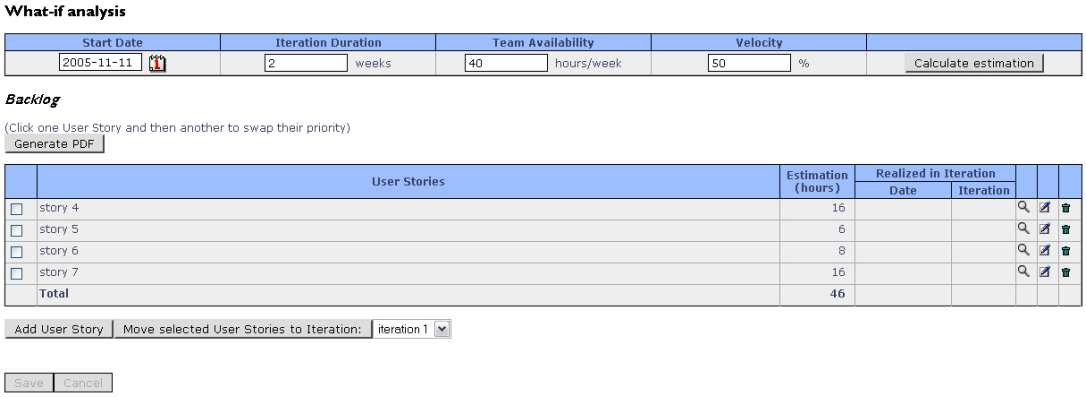


Figure 4.1: Backlog page in PPTS

The what-if option on this page is of especial interest for planning, especially on larger projects. This option calculates how many iterations will be needed to complete the estimated stories in the backlog based on the number of hours available and projected velocity. This provides an easy way to figure out how many iterations need to be planned for a project. It is annoying that the user needs to input a number of these values. Since they are available in the database. Especially the number of hours that people are available each week. This argument does not hold for velocity, since it is assumed that at the start of a project no historical velocity information is available.

Releases

Releases are at this moment not supported by PPTS, this means that iterations fall directly under a project. It also means that releases can't be used as milestones in a project. This is an issue when planning long-term projects that might have multiple releases. For small projects however releases are usually not necessary. Releases add an additional level of planning and tracking suited to long term projects, this would be a valuable addition to PPTS.

Iteration

Iteration planning in PPTS is rather confusing, mainly because the user needs to access a lot of pages to get feedback and move stories into an iteration. User stories can be moved into an iteration either from the backlog page or from iteration planning page. Unfortunately if it is done from the iteration page the user has to switch between a list of stories in the iteration and the list of backlog items. The user can't view both at once and move stories between the two in an easy way.

When stories are added to an iteration the only metric on the page shows how many hours the stories are estimated at and the total number of estimated hours. This information is useless if the user can't see how many resources, in person hours, are available for the iteration. To see this information the user needs to switch to either the burn-down page or the metrics page. both pages give an overview of the available resources per iteration.

Stories and Tasks

PPTS works with user stories as described in the chapter 2, these stories denote pieces of functionality and are subdivided into tasks. Tasks describe the steps needed to implement the functionality described in a user story. In PPTS user stories can be added on the backlog page, see figure 4.2, or the work breakdown structure (WBS) page. When a user creates a user story on WBS page, it is possible to choose whether to place it in any existing iteration or on the backlog. User stories created on the backlog page can only be added to the backlog and have to be added to an iteration afterward.

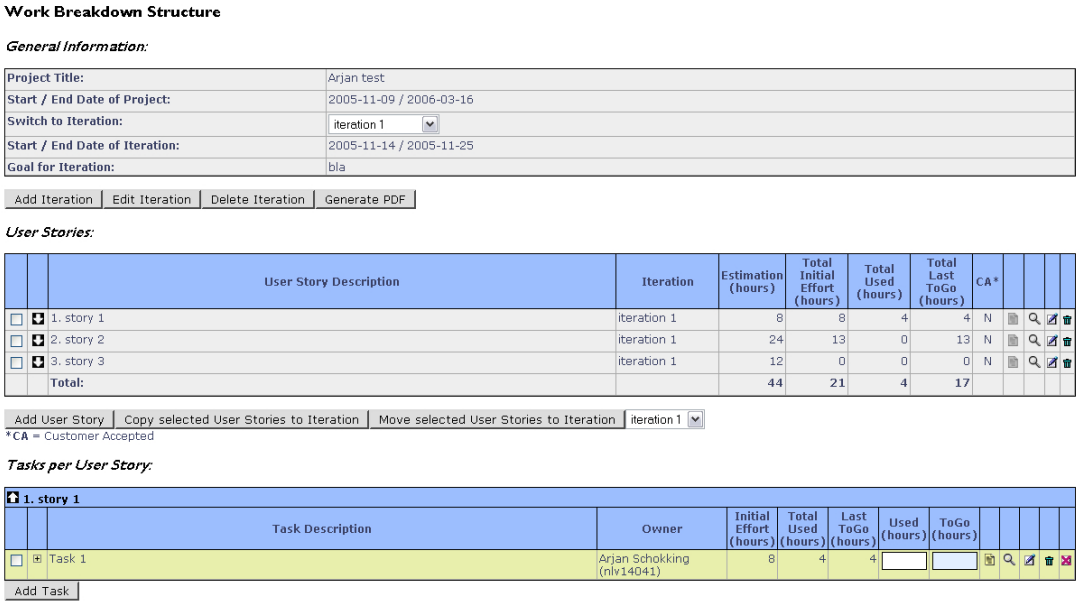


Figure 4.2: Work breakdown structure page of PPTS

Also interesting to note is that user stories can only be prioritised on the backlog. This is correct according to the Scrum methodology but does constrain users in what they can do with the system. On the subject of planning the only thing to be noted here is that tasks have their own estimates, which are usually better then estimates of user stories. Tasks are smaller pieces of work and thus, usually, easier to estimate. The rest of the functionality in PPTS regarding tasks belongs in the tracking section of this chapter, and will be discussed there.

4.2.4 Tracking

Tracking allows developers and managers to see how projects are progressing and is very important in agile methodologies. This is because, with few preconceptions about how a project will turn out, there is a need to control the course of the project as it develops. To do so people need to know what is going on in a project. This is where project tracking comes in.

Projects

In PPTS there is no tracking of anything larger then an iteration. This means that neither projects nor releases can be tracked effectively. As explained before PPTS has at this time no

way to define releases. This is a definite lack in the tool and adding release tracking should be a primary consideration.

Iterations

The tracking features of PPTS centre on the iteration. There are a number of pages that can be used to assess the status of an iteration. The first is the WBS, the page where an iteration is planned but the information here is not very easy to interpret since its in a numerical format and not very concise. A better page is the status report page shown in figure 4.3.

User Stories:

	User Story Description	Esti-mation	Initial Effort	Week 46 (14-Nov)		Week 47 (21-Nov)		Total Used	User Story Status
				Used	ToGo	Used	ToGo		
⊞	1. story 1	8	8	0	8	4	4	4	50%
⊞	2. story 2	24	13	0	13	0	13	0	0%
⊞	3. story 3	12	No Tasks defined yet for this User Story						
	Total	**44**	**21**	**0**	**21**	**4**	**17**	**4**	19%

Tasks per User Story:

⊟ 1. story 1

Task Description	Task Owner	Initial Effort	Week 46 (14-Nov)		Week 47 (21-Nov)		Total Used	Task Status
			Used	ToGo	Used	ToGo		
Task 1	AS	8	0	8	4	4	4	50%
Total		**8**	**0**	8	4	4	**4**	50%

Figure 4.3: PPTS Status report page

On this page all user stories are shown for the selected iteration and all tasks per user story. A numerical indicator and a progress bar depict how much of a user story has been completed. This gives a basic, but good, overview of how a iteration is progressing. However the time is only shown on a weekly basis so the time scale is small. A more detailed view of the progress of the iteration is given on the burn-down page. This holds the burn-down graph, which shows the work remaining for each day in the iteration. For these numbers the estimates of tasks are used.

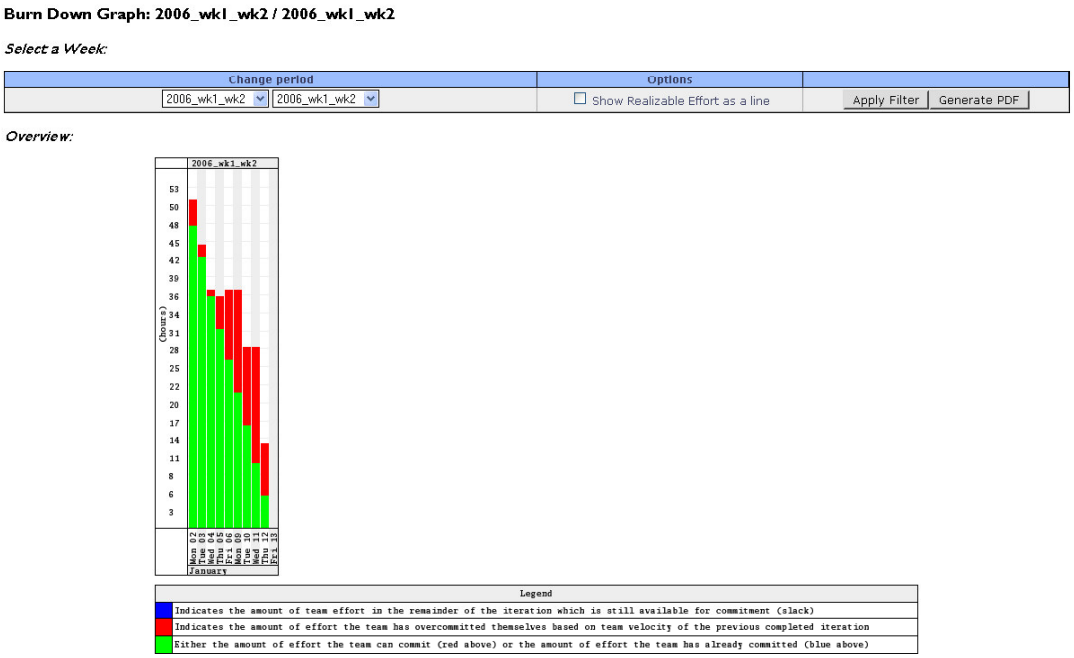
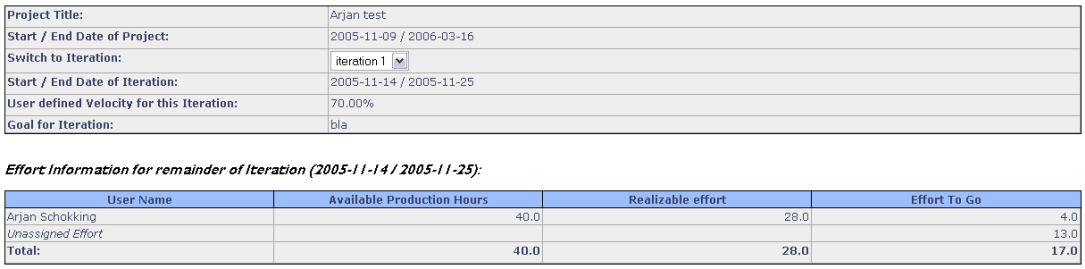


Figure 4.4: PPTS Burn-down page

This graph, shown in figure 4.4, shows if an iteration is going to finish on time and if there is too much or too little work planned for the iteration. In either situation corrective steps can be taken in consultation with the customer. Adding or removing stories can change the total amount of work for an iteration. On the metrics page, shown in figure 4.5, some more numbers are given that relate to the velocity of the previous iteration and the numerical representation of the burn down graph. It does not add much in addition to the burn-down graph and could be incorporated into the graph page.



Project Title:	Arjan test
Start / End Date of Project:	2005-11-09 / 2006-03-16
Switch to Iteration:	iteration 1
Start / End Date of Iteration:	2005-11-14 / 2005-11-25
User defined Velocity for this Iteration:	70.00%
Goal for Iteration:	bla

Effort Information for remainder of Iteration (2005-11-14 / 2005-11-25):

User Name	Available Production Hours	Realizable effort	Effort To Go
Arjan Schokking	40.0	28.0	4.0
Unassigned Effort			13.0
Total:	**40.0**	**28.0**	**17.0**

Figure 4.5: PPTS metrics page

Stories and Tasks

User stories are more important to planning then they are to tracking. Since user stories are the main unit in which work is defined for planning but tasks are the most accurate level at which tracking occurs. However, on the status report page, there is an overview of how all stories and tasks are progressing. Unfortunately this is not linked to how much time remains in an iteration. While it can be concluded that a story is not finished yet, how much time it will take to finish is harder to determine.

The main unit of tracking and estimation is the task. The estimates of hours spent on a task, and of how much more time a task is going to take are the main units on which progress is defined and tracked. Thus, it is essential that developers fill in their hours on the personal page regularly. If users do not do this regularly the team's information about the status of the project will be based on inaccurate or incomplete data. There exists a tension between providing current up to date data and flexibility in filling in the work done on tasks. Whether the user should be forced to fill in their hours each day by some mechanism is outside the scope of this report and is up to Philips. Some form of feedback that the user still needs to fill in their hours would be a recommendation though.

4.2.5 Other

There are a few items that do not fit into the above categories and still need to be mentioned. They are discussed here.

Transparency

The PPTS tool is easy to understand if the basics of Scrum and extreme programming are known. If they are not, there is no user manual or other guide that will help the user understand the program. While a good understanding of agile programming methodologies and Scrum in particular will allow a user to work with PPTS, the complete lack of documentation other then the installation guide is a big problem. This is especially true if there is no one around to answer questions about PPTS.

The transparency of the tool in regards to normal usage and how different parts of it interact, or in other words: "what is happening" is confusing at first. This is mainly due to information being on more then one page and the need to jump between pages to accomplish common tasks. In this aspect the tools transparency is bad.

Help features and documentation

Documentation for PPTS is almost non-existent. There is a brief installation guide, which explains the steps the user needs to go through to get the software running. But anything other then this does not exist. A partial solution to such a problem could be in the use of tool tip help, unfortunately that is not included in the system either which leads to the occasional guess as to what an unlabelled button will do. Because of this it is important to have either someone explain the system or have a good working knowledge of agile processes to enable educated guesses. Of course good documentation would be a lot better then guesswork though.

The metrics side of the system is better documented with clear labels on the burn down graph and a tool tips showing additional information for each day in the graph. But it must be concluded that the system is badly documented and should have at least a quick tutorial and an explanation of the basic functionality.

Responsiveness

Responsiveness of PPTS is good with a notable exceptions having to do with the generation of big graphs. Such as the Resource Allocation graphs. This is due to an inefficient implementation of the graph building, which causes a heavy load on the SQL server due to too many queries. The issue is being worked on however and the new versions of PPTS that have been released during this study are much better then the first one was. However since PPTS is a web application there always are problems associated with that architecture.

Bug reporting

PPTS has no inherent bug tracking mechanism. It does link with either a Mantis or Bugzilla database and lets the user import bugs into the system as stories or tasks. This is not an optimal solution since interface issues are always present when working with outside programs. But it is better then nothing.

4.3 PPTS and Agile Software Development

PPTS supports the agile development practices at SES.

Though SES uses the XP@Scrum methodology, which is a combination of both agile methodologies. XP is more concerned with engineering practices and Scrum deals more with management and organizational issues. PPTS focuses more on the aspect of Scrum from XP@Scrum then on the engineering practices embodied in XP. This is logical since it is a project planning and tracking tool.

4.4 Summary

Here follows a quick summary of the main points in PPTS that need attention. This provides a focus for the review of other tools. Since it is worthwhile to look for solutions to these problems in how other tools have implemented these features in comparison to PPTS. The main points in PPTS that deserve attention are:

- Planning :
  As stated in the planning chapter, the planning of projects and especially the addition of users to projects can be improved upon. The complete lack of release planning should also be addressed if it is needed in Philips. Iteration planning should also be changed since it is too cumbersome in it's current implementation.
- Tracking :
  More tracking on a higher level of abstraction, either project or release based would be an improvement. However, release features would have to be added to PPTS first. Displaying more of the information that is available in the database in a clearer fashion would be useful if it is possible.
- Usability and transparency : If the items mentioned above are changed, disregarding for the moment what form that implementation should take, both usability and transparency could be increased. Since a clear and simple way to plan and track projects would lead to a better intuitive understanding of what is happening in the tool and thus, a higher degree of transparency. The addition of the extra level of planning provided by including releases could also improve both these factors especially in large projects.

- Documentation : Good tools should have good documentation. Since this tool is mostly used inside Philips it is not hard to ask someone what something does. But having a clear and descriptive manual for PPTS would save everyone time and effort since the time needed to clarify features would be less. If the tool should find more use outside Philips itself and if it need to be deployed further inside the company then more documentation would be necessary.

Chapter 5

Results

This chapter details the results of the comparison between the different tools. The list of basic functionality in each tool, compared to the others, is given in a product comparison matrix included in appendix C. In this chapter the features that were not easy to include in the comparison matrix are explained in more detail. This list was created to identify interesting features, both good and bad, which could lead to improvements in PPTS. As such this list is not complete for all features in the tools. Only the features, which are interesting for further review, are included. Some tools will have the same interesting features to incorporate. These will only be mentioned the first time they are encountered.

5.1 Introduction

The following section, which discusses the individual tools, is quite extensive. A short overview of good and bad features for each tool is included in appendix D. For a quick overview it is recommended to read this first and then look up any specifics in this chapter.

5.2 XPWeb

XPWeb is an open source tool for enabling the agile development process. It is based mainly on the Extreme Programming agile methodology (see chapter 2). XPWeb's main features can be viewed in the product comparison matrix in appendix C. The good and bad points in XPWeb, which fall outside the scope of the comparison matrix, follow.

5.2.1 General remarks

The next section details some general remarks about XPWeb.

Good

XPWeb is a LAMP (Linux, Apache, MySQL, PHP) application, and as such relatively easy to install on a number of different platforms. XPWeb also allows the user to initialise its database from the WWW interface, which is easier then configuring it manually with MySQL. This is a nice feature, although not very consequential.

The help section of the XPWeb is excellent and explains everything the user needs to know about using the tool. This is one of the better features of XPWeb. The help functionality is

fully integrated in the interface and provides explanations on both agile methodologies and XPWeb's functions. The help functionality is structured so that each page in the tool has an accompanying help page. This structure is very intuitive.

bad

A lot of functionality in the tested release of XPWeb was broken. The sort feature as well as the calculation of end dates for iterations did not demonstrate expected behaviour. In fact both seemed broken.

All basic functionality that can be expected from an Extreme Programming perspective is in the tool. Although, whether it is well implemented is open for debate, this is discussed in the next section. The only thing missing in XPWeb is release planning.

5.2.2 Planning

XPWeb includes a calendar, see figure 5.1, which shows the start and end dates of each task. It is assumed this is why tasks have start and end dates. Since a task is usually only described by a title, an estimate and the number of hours left till the task is completed. The calendar view might be useful for micro management of tasks but quickly becomes confusing when a large number of tasks are displayed. Each task is also assigned a random colour, assigning a colour to a task according to which story it belongs too seems more reasonable. The functionality that the calendar provides seems to support that XPWeb is a tool for monitoring project slippage. However, burn-down charts are a much clearer way of visualizing this.

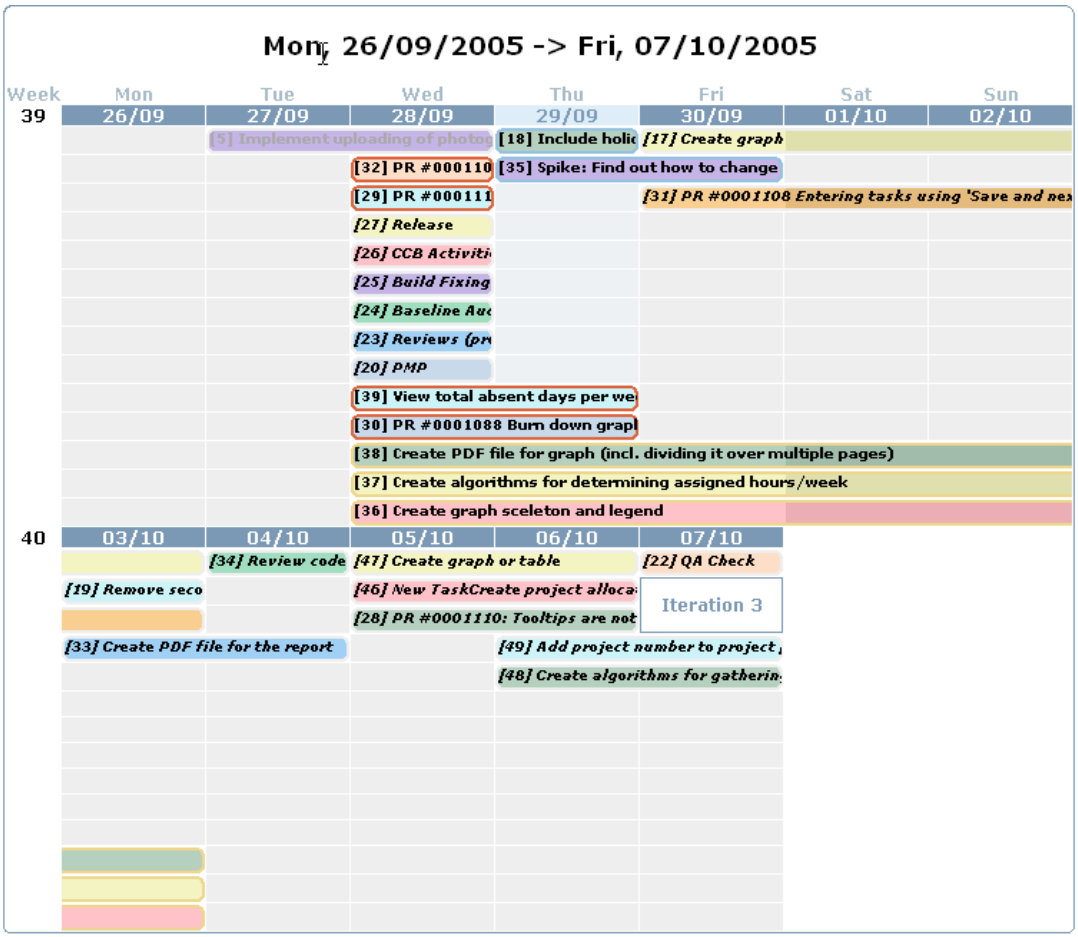


Figure 5.1: XPWeb calendar page

5.2.3 Tracking

To effectively use a planning tool for agile development a user must be able to keep track of the time spend on a project, and the time the user still needs to finish the project. Since this is the most basic task in many tools, it is important that this task is properly facilitated. However the number of actions needed to do this in XPWeb, namely: Open the project page, open an iteration, open a story, open a task, click modify, fill in the numbers and finally hit save, is excessive for performing a common activity.

The question, why start and end dates for tasks were included at all, was asked before. In an iteration a developer start working on a story performing its tasks in sequence and marking them done when they are completed. Only the hours spend on the task and the time still needed until it is completed are important. These two figures allow the developer to estimate whether the story will be completed on time or whether it will slip. The most logical reason for the inclusion of start- and end dates for tasks is that they were needed to implement the calendar page.

5.2.4 Interesting to Incorporate

Below is a short list of features that are interesting to incorporate in PPTS.

- Integrated help functionality in the form of a help button on every page, which automatically sends the user to the section in the manual, which covers the currently active, item or page.
- While the installation is done manually, XPWeb uses an automated script when a user logs in for the first time to initialise the database. This feature might be nice to put in PPTS, but it becomes more complex as more databases need to be supported.

5.3 ExtremePlanner

ExtremePlanner (EP)is a commercial tool and it is easy to install since it comes with a windows installer. The tool does not follow a specific methodology but is a general agile planning tool with elements of XP and Scrum. By including releases ExtremePlanners scope is slightly larger then that of XPWeb. However, the extra features added with the inclusion of a release are minor. They include setting a date for a release and adding user stories to a release. The user cannot assign iterations to a release. This seems contradictory since an iteration is the highest hierarchical item next to a release. In other words: User stories should be coupled to an iteration and iterations should belong to a release. The method ExtremePlanner has chosen, which ties user stories to a release, seems counter intuitive.

5.3.1 General Remarks

ExtremePlanner has a number of nice features, which are worth discussing. The ones that don't fall under tracking or planning will be discussed in this section.

Good

One interesting feature is the ability to import stories and export stories and tasks from and to Excel. This is useful when working with spreadsheets or when entering historical data into the system. However, when importing stories it creates extra work when compared to inputting the user stories directly. The export feature is more useful. It allows the user to make printable documents to show to management or the customer when no computer or Internet connection is available.

ExtremePlanner has extensive sorting features. The tool can sort stories and tasks by almost all common characteristics of an agile project including: Releases, iterations, priority, risk, value, name and more. EP also allows the user to save a custom setting as a default sort when viewing a page that contains a list of items.

The interface is very clean. However, in some cases it can become a bit cluttered when there are long lists of stories or tasks on a page.

5.3.2 Planning

Good

EP allows the user to perform all the expected actions, like creating user stories, iterations and tasks. It also allows the user to track the status of projects by entering estimates and

the times spend on tasks. Although the following section contains a few remarks on how this functionality is implemented.

Bad

How the release functionality is used in this tool can be questioned. ExtremePlanner only allows the user to set a date for a release and assign stories to a release. No additional functionality is included to do anything other then these two things. Release planning options, an overview of what iterations and user stories are in a release and a visual representation of the progress in a release would be needed to make this feature useful.

EP provides little help to the user when planning a project. There is an overview of the total number of hours in a certain selection of tasks or stories. However, there is no velocity figure or historical data to help the user select a starting velocity. There is an overview of velocity with as unit the number of user stories completed in a given interval. But without data about the size of each use story in ideal hours this information is meaningless.

The illogical placement of editing options becomes apparent when planning. For example: A user cannot add or remove user stories from the iteration page or the edit iteration page. It can only be done from the modify story page, which is very counter intuitive. There are a number of these fallacies in the program. They do not cripple the program but are very annoying and make the tool unpleasant to work with.

5.3.3 Tracking

Good

The iteration overview page, see figure 5.2, is excellent. It gives the user a clean overview of what is done and what is in progress or not started yet on a story-by-story basis for each task in each story.

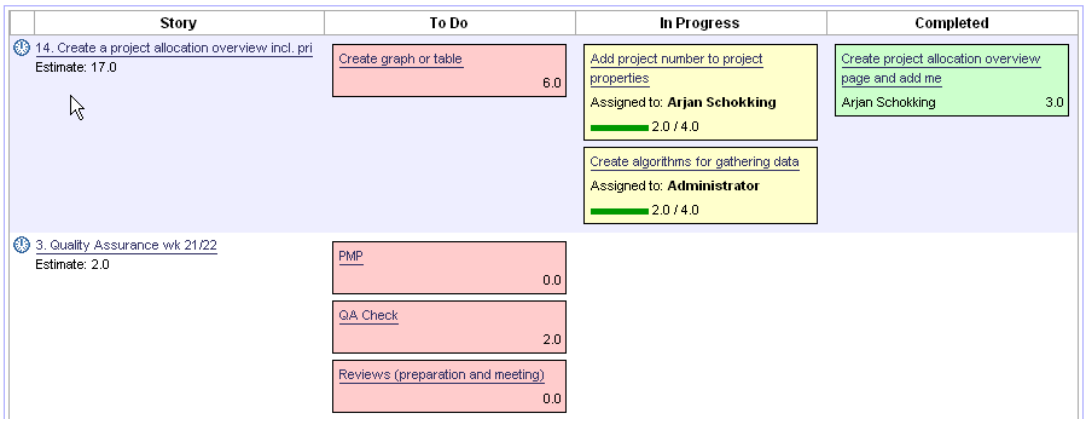


Figure 5.2: ExtremePlanner iteration status page

Extreme planner also features a summary page, which serves as the entry point of the tool. It has a short overview of the tasks and stories completed on a release, iteration and user basis.

The metrics included in the tool are a project burn-down graph, which shows how each iteration went, and an task estimation accuracy graph. This is a bare minimum and more useful graphs were present in other tools.

Bad

On the iteration overview page it can be debated if ExtremePlanner uses the best way to start a task. EP gives a task the status of started the moment a developer is assigned to the task, even if the developer has spend no time on it yet. If the starting point of a task is a compromise between when a developer is assigned, and when time is booked on the task. It would be closer to reality. The same sort of problem occurs when a task is completed. This is done by manually setting a flag for the task to completed. A more natural option would be to have EP set a task to completed when the Togo of the task reaches zero hours. This is a more intuitive method and it requires less user interaction.

The iteration overview should be compact. This is not the case in EP, since in a typical project where an iteration might have a dozen or so stories the overview can become a very long list. This defeats the purpose of having a overview that is understandable at a glance, since the user needs to scroll up and down all the time.

5.3.4 Interesting to Incorporate

- Some improvements to the iteration overview page. Proposed is something like the ExtremePlanner version but more compact with some other criteria for which column a task is in, and more information included on the page
- Releases and release based metrics like the burn-down graph. Preferably with a more stable unit of measurement
- Sorting features and sorting profiles for all work items like tasks and user stories. The user should also be able to set a sorting template
- Start at a comprehensive project summary page when a user logs in. Such a page would need more information on it and better links to other sections of the tool then is the case in ExtremePlanner

5.4 Rally

Before starting with the results of Rally there are some things that are different from the other tools and they will be discussed first. Rally is a hosted solution with a restricted preview feature. This made it hard to use the data set described in chapter 3. Therefore, this tool was tested with the data that was available in the Rally trial. Since Rally includes a full project in its preview, which seems to be similar to a normal IT software project, and the testing is based on the features and functionality of Rally this should have no impact on the analysis.

Rally and VersionOne, the tools discussed in the next sections, are commercial tools and are geared to facilitate larger projects then the other tools that were analysed. They adhere to a stricter path through the tool when planning and estimating projects, which is especially true in VersionOne. This leads to a number of extra features and functionality, which are not directly related to the Scrum or XP methodologies. These additional features are not included in the comparison matrix but will be discussed briefly.

5.4.1 General Remarks

Rally is a tool that allows the user a lot of freedom and options to plan projects. Unfortunately it seems that the developers of Rally wanted to include too much in the tool, which led to a unclear and too complicated structure of internal work items and deliverables. Some of the names that are used by Rally don't match the ones used in agile methodologies either. This makes the tool a lot less intuitive to use then it could have been. A more detailed explanation of these problems follows in the next sections.

5.4.2 Extra features

Rally has a somewhat different structure concerning the break down of work then the previous tools. Rally does have projects, releases, iterations, story cards (user stories) and tasks. But it also has a secondary structure which Rally calls the feature breakdown structure. The top level of this hierarchy is called the workspace. This contains features and defects. Features contain use cases, requirements (functional and non-functional), test cases with their results and defects. This parallel breakdown structure maps onto the breakdown structure that was mentioned first by using story cards to schedule pieces of work. These story cards then usually link to a feature or requirement or a defect.

Features, use cases, requirements, test cases and defects are of course well known terms in software development, and there is something to be said for including them in the tool. It allows the tracking of bugs and defects to be handled in the tool instead of in a separate program. It also gives the user more levels of details on which to describe work items.

However, on the other hand it complicates the structure of the program and leads to unexpected behaviour compared to what the user would expect from the basic Scrum hierarchy. This can be confusing and leads to a steep learning curve for the program. Since this is a comparison in which the time spent on studying each tool is limited it is impossible to say whether the structure used by Rally is as confusing as it seems at first or whether the added complexity is outweighed by the extra functionality it offers. However it is possible to conclude that the start-up costs of this tool will be higher then with the earlier tools due to the higher complexity.

Rally also has a general search feature, which lets the user search all work products for a certain keyword. This is useful when a user needs to find a specific story or something else in a large project.

5.4.3 Planning

The Rally tool allows for a extensive amount of planning to be done beforehand. Although the logical steps in the planning process are not especially clear from the interface. The tool supports planning of multiple projects. Each containing iterations, stories, tasks and defects and tests. Guidance on how to run through the planning process is given in the documentation and a number of tutorial movies.

Good

Rally has planning features on the level of releases and iterations. Both feature a drag and drop interface, which lets the user drag stories and features to an iteration or release. Rally also has indicators, which show the remaining amount of resources for an iteration, and the balance of planned work versus resources. This gives the user real-time feedback on the planning process so that the right amount of work can be planned for each iteration. Additionally, the user is

able to move any number of stories and features from one iteration or release to another via select boxes.

Bad

Releases are planned using features, use cases and requirements. Iterations are planned using story cards, which the tool generates automatically for each feature that is added to the backlog. This is a bit confusing for users who are not completely familiar with the program yet. The previous tools, which only used the standard work breakdown structure as described in the extra features section of the Rally tool, are a lot more intuitive in this aspect.

A more logical workflow in the planning process would also be better. The pages where the planning is done are named intuitively but they are not arranged in a fashion that clearly indicates a start to finish planning process. This especially noticeable when the user starts planning a project or iteration.

5.4.4 Tracking

Rally let's users track just about everything in a project. Most of the tracking information is displayed in text and numbers complemented with a number of graphs.

Good

Rally features overview of iterations, releases, stories, defects and tests. There are also lists of all requirements, features and use cases that have been specified. All of them can be sorted by all their attributes. Stories and tasks have a graphical status-indicator, which indicates whether they have started, been completed and accepted, or are blocked in one of these stages.

Rally produces quite a few separate overview pages. This does allow the user to view all the relevant information, even if some of it might appear in several places.

Other nice features are the cumulative workflow graph and the burn-down graphs. The latter are available for both Iterations and Releases. Both show, in addition to the normal information they contain, which parts of the work are being: planned, worked on, or accepted. this gives an additional level of detail to the graphs. The downside of these additional details is, that they make the graphs more complex. But this is not a serious problem.

Tracking on a individual level is implemented on the personal home page that a user starts on when he logs in. This personal homepage shows the users currently assigned tasks and any open defects. It also provides a list of any work items that have changed since they last logged in and if those changes impact any work that is assigned to the user.

Finally, there is a chart, which shows a time-line for the project. It includes releases and iterations and is very useful for getting up-to-date quickly on the projects status.

Bad

As was already stated in the section above, the various reports are scattered over a number of pages. Having them more centralized for easy viewing could be a better approach.

5.4.5 Interesting to Incorporate

- A graph of the projects burn-down and one of the acceptance of user stories

- A personal starting page for each user that shows up when users log in. It should have relevant information like: tasks assigned to a user, bugs assigned to a user, and changes by other team members, which impact a users tasks
- A better and clearer planning interface
- A planning status for iterations and releases, which states: resources remaining, described in person hours coupled with velocity. This velocity can be based on historical data or a best guess if there is no historical data available. This page should also include the balance between resources and work that has been planned. Furthermore, it should automatically update as stories are added, removed or moved to another iteration or release.
- The ability to split user stories when in the iteration planning phase of a project, and the ability to move part of the split story to another iteration.
- Releases and release planning features, these should work the same way that the iteration planning interface does.
- Move to a slightly smaller font to increase the available space on the screen.
- A better page layout.
- Horizontal tabs for menu items instead of vertical drop-down menus.
- The feedback on most of the users actions should happen on the screen where the action was initiated. This stops the need to look at multiple screens to see what effect some action had.

5.5 VersionOne

VersionOne is the second large commercial project planning and tracking tool that was evaluated. Like Rally, its set-up is structured in a better manner then that of the first two tools that were discussed. This makes planning projects in a coherent fashion easier. The users also need to have less knowledge of the tool beforehand.

5.5.1 General Remarks

The number of pages in VersionOne is markedly smaller then the number of pages in Rally. They also provide a better overview of what information can be found where. Finding the correct information was sometimes bothersome in the Rally tool.

VersionOne has a nice and clean menu layout, with the four main items: Projects, my home, reports and administration. These menu items are always available at the top of each page. The rest of the menu items are displayed below this top-level menu. The only obvious implementation flaw was: that a number of charts and overviews are linked to their objects, like iteration and release burn-downs. These overviews cannot be directly accessed via the reports section, which is the most obvious place for the charts and graphs. Once the location of these views is known it is less of an issue. However, it would be better if this information was also accessible from the reports section even if it is only through some extra links.

5.5.2 Extra features

VersionOne can be used both as a service hosted by VersionOne and as a stand-alone site that is run on a local web server. This provides more flexibility when compared to Rally, which only runs as a hosted service. Of course this could be an issue for projects that deal with sensitive information or technology. In principle a hosted solution would allow the hosting company access to all the project data. With the amount of corporate espionage around in the world, especially in research environments, a lot of companies might be reluctant to store data anywhere but on their own servers.

5.5.3 Planning

Here the planning features in VersionOne are discussed.

Good

VersionOne has a very structured way of walking the user through the set-up of a new project. It guides the user in creating a project, release and some iterations, so they end up with a planned project. This is especially useful for newcomers to the tool and to agile methodologies. After these basic steps, the next entries in the menu deal with filling the backlog and planning the iterations and sprints. However, there should be a better link between the last step of the set-up process and the start of the planning process.

The structured way of working mentioned above is entirely supported through the layout of the menu. If a user has already worked with the tool they can do the project set-up in almost any order they like. So, in addition to the structured set-up, advanced users can use the program as they see fit.

Bad

VersionOne does not give a lot of feedback in the planning process. No historical data or the ability to set the expected velocity, and see how much time some user story is going to take, is present. Neither are resources, coupled with a velocity, used to predict an acceptable workload for an iteration. This is something that should be present in a program that is as professional as this one. What information the tool does provide is on other pages, which means skipping back and forth during the planning process. While it is not a disaster, it is annoying.

All linkages between releases and iterations in VersionOne seem to be implemented on the user story level, which means that iterations are not directly attached to a specific release. This seems contradictory but was probably done due to implementation issues. There is a possible opportunity here for automation of time planning tasks such as planning iterations in a release and what-if analysis. A what-if analysis can be based on the current backlog estimates, the iteration length, release length and available resources.

5.5.4 Tracking

In this section the good and bad highlight of the tracking features in VersionOne are reviewed.

Good

VersionOne has a lot of tracking graphs and metrics, which show the projects current status. On the start page of VersionOne is an overview of current projects and how they are progressing. This is a progress bar, which indicates the percentage of the backlog that is finished.

From the start page the user can access a number of graphical metrics about the project. Additionally, almost every page has numerical metrics about the progress of the its contents. For example: on the sprint tracking page the sprint and task estimates and the done and Todo numbers are shown.

Bad

There is a lot of tracking data in the tool, and it is available almost everywhere. This can lead to not seeing the forest for the trees. Having the tracking data summarized on one page and reduced on other pages might be a better idea.

5.5.5 Interesting to Incorporate

- The planning interface is well implemented and it is worth looking at
- The way the menu structure points a user in the right direction while setting up and planning a project
- Some way of indicating how far along a sprint is

5.6 Xplanner

Xplanner is another open source tool that is easy to use but has limited functionality. It is well suited for keeping track of small projects but does have a few weak points, which will be discussed below

5.6.1 General Remarks

Someone with a good amount of IT knowledge should install this tool. It is not very easy to get running. Xplanner runs as a web-service under tomcat, a web service server, and needs a number of other applications installed, in addition to the program itself, to function.

This tool is simple and limited in the features it provides the user to manage projects. This makes for an easy-to-use program with few difficult options. It also leads to less flexibility then bigger tools. Mostly smaller sized projects should be planned with Xplanner. In bigger projects it is useful to have release support, which this tool does not.

The program does not have a back-button, which means users have to use the back button of the browser itself. This is usually a unwise thing to do since values can get overwritten.

Users can add notes to each of the overview screens to notify other team members of things that might block progress or other problems. This is a nice feature when working in an environment where not every team member is readily available.

5.6.2 Extra features

This tool let's the user set a busy flag while the build is being integrated. It is an interesting feature that is not present in any of the other programs but whether it is truly useful is unclear.

The tool gives the administrator the ability to send warning emails if developers forget to fill in their hours. This is a new feature, which was not present in any other tool. It can be useful when people forget to fill in their hours. Plus, if users do not fill in their hours regularly the project tracking becomes less accurate.

5.6.3 Planning

Planning in Xplanner is basic. The user creates a project and an iteration, then the user creates use stories in that iteration. Finally, the user adds tasks to each user story. The system does not feature any planning metrics or statistics other then showing the total estimate of all the user stories in an iteration.

Good

The tool is easy-to-use and clear about what items belongs where since projects, iteration, stories and tasks are displayed in a nested format.

Bad

Xplanner provides no feedback to the user, other then a total estimate, when doing planning. Xplanner does not display resources or any suggestions about how much work can be planned for an iteration. Neither is there any way to set an iteration's velocity.

5.6.4 Tracking

The tracking functionality in Xplanner is more extensive then the planning features. The tracking section of the tool features charts, metrics and other information about the status of an iteration. Most of the tracking statistics are based on iterations. Some metrics about stories and tasks, concerning the amount of work that has been completed and how much effort is left, are also included.

Good

The information the tool provides to users is easy to find. From the iteration overview page the user has access to the metrics, charts or accuracy pages.

The metrics page is interesting, it let's the user view how much work was done singly and how much was done in pairs, in addition to showing totals. This is quite novel and gives the user a good idea of how much pair programming is being done.

In accordance with the last section, when a user inputs hours spent on a task it is possible to choose whether the task was done alone or was worked on with another person. This focus of Xplanner on pair programming is nice and is lacking in many other tools.

The charts that the tool provides are a basic burn-down and a completion graph.

Bad

Nothing about the tracking in Xplanner seems especially flawed.

5.6.5 Interesting to Incorporate

- Keeping track of pair programming in some fashion.
- Alerting users to the fact that they still need to fill in their used time for tasks
- Adding notes to pages.

5.7 TargetProcess

TargetProcess is situated between the smaller agile tools and the big ones such as Rally and VersionOne. This is good since it gives both the benefit of being able to do smaller projects, while ignoring the functionality that is not used, and being able to handle larger projects. This is however also true for VersionOne, and, to a lesser extent, for Rally. But both of those have more upward scalability.

5.7.1 General Remarks

The first thing that is noticeable about this tool is its clean interface. It has few bells and whistles and provides a very clean overview of the necessary information. However, TargetProcess is not structured very well. This is especially true for the menu layout and overviews. However, with a bit of practice this should not pose a problem to users. Although it does detract from the intuitive use of the program.

5.7.2 Extra features

TargetProcess has some nice extra features related to communications. It has a section for posting files on the system and, and this is pretty unique in the compared tools, there is a messaging page where developers can relay additional information to the rest of the team.

Another nice feature is the information about the workload per team member, which can be seen on the team page. This gives a clear picture of how much effort each member of the team has available in the current iteration, and how much work they have been assigned. It would be nice if this status report changed along with the current date, or in other words, also shows the possible effort left. This has however not been implemented possibly because this status also lets the user set the amount of effort available per week, which is a constant.

5.7.3 Planning

The planning section of TargetProcess works the same way as most agile tools. First the user defines a number of user stories with initial estimates and creates either a release with some iterations or just some iterations. Then on a different page the user can assign the user stories to iterations, and if applicable, a specific developer.

Good

TargetProcess is the first tool that has an automatic iteration-planning feature. It assigns user stories to iterations using size, risk and business value parameters. A couple of things can be noted here: First, the way that agile development usually works does not really support this feature. The customer is the one that decides what user story should be worked on first, and automatic planning counters this principle. On the other hand if all the parameters are set correctly it can save the user a lot of work when dealing with a large number of stories and iterations. The second note with regard to this feature was that it did not seem to work but it is assumed that the fault in that lies with the testing and not the program since the amount of testing time available per tool in this project is limited.

Another nice feature, which should be in all tools, is a counter that keeps track of the planned effort against the available resources in an iteration. The velocity set for the iteration denotes the available resources and the planned effort is denoted by the sum of the estimates of planned user stories.

Bad

Annoying in this tool is the way commands and menus are structured. The menus are not always logically structured. The sequence of commands that a user expects while planning is not always clear. There is not enough focus on the control-flow though the planning process. Also, options to, for example: Add a user story, are located in a sidebar that is always available. Instead, users expect such a feature to be on the user story page where this action produces a result. Being able to add stories on almost every page is nice, but having the option available just where it is expected is much nicer.

The user is able to set the velocity for an iteration. However once the iteration ends the program does not use the historical information available, the velocity at the end of the iteration, to determine the velocity for the next iteration. Instead, the tool has the user fill in the velocity again. While this is not a big problem, it would be nice if the tool determined a possible velocity for the next iteration based on the past few iterations.

Another thing, is the overwhelming amount of information given in the screen that lets the user assign user stories to iterations. This is so much that a user story quickly occupies three to five lines of screen space. If the user must choose some user stories from amongst a dozen or more this becomes very confusing. The user needs to scroll all over the screen to find the user stories he needs or wants to add to a given iteration.

5.7.4 Tracking

Tracking in TargetProcess is, compared to the big tools like Rally and VersionOne, pretty basic. But the basics it has are well done and provide the user with the information they need about the projects status. Additionally, there is a complete internal implementation of bug tracking and test planning. Both of which are easy to use and intuitive.

Good

The first nice thing about tracking in TargetProcess is that the iteration status page displays the amount of effort realized versus the amount of time elapsed in this iteration. If more time, in percentages, has elapsed then work is completed the indicator indicates bad and if more work is complete, in percentages, then time has elapsed the iteration status indicated good.

The addition of messaging features helps with tracking as well. It allows the users to communicate information about the project, which falls outside the scope of the provided metrics. It also allows the user to express any additional information that they feel is important, and is best expressed in natural language. This is a powerful extra tool.

A quick overview, which is given at the bottom of the main page, shows the number of stories done and bugs closed the previous day. A nice feature, which in the case of a well run project can be quite motivating when a user starts his workday.

Bad

The different metrics and reports that TargetProcess uses for tracking progress are spread out through the tool. This can be a good feature when the metrics a user needs are present on the pages they use most often. But a manager or executive wants all these reports and figures on one page. This is lacking in TargetProcess.

The way time spent on tasks is recorded is rather cumbersome compared with other tools. In TargetProcess this works by clicking the add time link in the quick links or on the time keeping page. This lets the user add an amount of time and a description about what the time was spent on to a story or task, selected by a drop down box. Better implementations use a

simple field on a story or task overview list, which lets the user, fill in all their time spent for a day in a single go. The method this tool uses creates too much overhead for this common task.

5.7.5 Interesting to Incorporate

- A project status
- A figure, which shows the load of the different developers compared to the time that is available for work.
- The ability to have a messages page per developer, this helps internal and external team communications.
- An overview of status changes when compared to the previous day. Including: Number of tasks, stories and bugs closed and or added.
- When planning an iteration show the available and the iteration that is being planned.

5.8 Concluding Remarks

The purpose of this analysis is not to provide a general comparison of the different agile tools that were analysed. However, during the work on this analysis there has been interest from the agile community for a comparative review. So, in this last section of the chapter, there are some concluding remarks about each tool and what kind of use each tool is most suited for. This is, of course, partially based on a subjective opinion. Since each of the tools mentioned in this report has a trial available. Anyone that is using this report to make a decision about which tool to use is encouraged to try out the tools they feel best fit their needs and make a final decision based upon their own findings.

5.8.1 General Remarks

The reviewed tools can be sorted into three categories, based on the size of the projects that they are used to plan and track.

5.8.2 Tools for small projects

A lot of agile projects just have one team with a couple of members. These teams do not need a tool that supports massive development efforts and which offers a lot of unnecessary functionality. Many small teams actually use the simplest tools of all, cardboard, tape and a marker. While this work well for planning small projects there is something to be said for a software tool. If only that it allows the team to save the project history for reference at a later date.

When small teams do use a software tool. They typically do iteration planning, and use the tracking features to control the development of the software. With a burn-down graph and perhaps a velocity graph it is then easy and intuitive to track progress and notice any trouble before it becomes a real problem.

Most of the tools in the comparison fall into this category, namely:

- XPWeb
- Extreme Planner

- Xplanner
- PPTS

These tools, in most cases, distinguish themselves from their big brothers like Rally and VersionOne by keeping it simple, at least in concept. Of these tools only Extreme Planner uses release planning, and in that program its implementation is very marginal. Probably the most important things when using a tool in a small project environment is a good user interface, intuitive functions and a clear representation of the data. Especially everyday activities like entering hours spent on tasks and task assignments should take a minimum of effort.

5.8.3 Tools for medium sized projects

The second set of tools is those, which are appropriate for running medium sized projects. Perhaps more then one and also using more teams. There is actually only one tool that really falls into this category and another tool, already mentioned in the small projects section, which also qualifies. The separator in this case is a tool that has Release tracking but is does not have much extra functionality. This allows for long-term planning without bogging a project down with excess functionality, which is never used or is used but turns out to be irrelevant.

The tools that fall into this category are:

- ExtremePlanner
- TargetProcess

ExtremePlanner was also mentioned in the category of small tools and its size is between that of a small and medium tool. Both of these tools have a simple implementation of release planning and some extra features. But the release planning is very simple, adding the ability to define a release milestone and assign stories to it. But the tools lack the extensive planning and tracking features on release level that the tools suitable for big projects provide. These two tools are also usable for small projects but are less suited for big multi-team projects. For those bigger projects the tools discussed in the next section are better suited.

5.8.4 Tools for large projects

Finally there are two commercial tools, which are geared up toward managing larger and multiple projects, and multiple teams. These two tools are both commercial and as such have a cost factor associated with them. They do however provide the ability to do extensive release planning with separate backlogs for each product and release. Also they have good reporting and metrics included that enable managers to keep an eye on how their projects are progressing. The fact that these tools have more options and features then their smaller cousins also means that they have a steeper learning curve and are especially in the case of Rally harder to navigate. These are the two big tools:

- Rally
- VersionOne

For users who are in big organizations using the smaller tools for cross team projects is not advisable. They should have a look at the trial versions available at the Rally and VersionOne websites and determine which one they feel fits their needs the best.

5.8.5 Concluding

The main point in the decision about which tool to use is the scale of the project, the situation and personal preference. Each of these tools works differently and focuses more on some aspects of agile methodologies then on others. Since each project probably uses a subset of agile practices as well, it is useful to match these to the tool as much as possible. For this purpose the comparison chart in appendix C can be used. With the comparison chart as a guideline it is strongly advised to test the trial or full version of the tool first before starting to use it in a larger context.

Chapter 6

Conclusion

This chapter has some final thought about the research thesis and some ideas about additional research that is of interest to the agile community.

6.1 Conclusion

Agile methodologies are exciting. They are exciting because they promise something better in the realm of software development then what is currently available. Agile methodologies might not be a silver bullet for software development, but they codify a number of practices that work well in specific situations. If the situation and environment are given appropriate thought in the decision on which methodology to use, these methodologies can be a great asset for structuring the development effort.

Agile methodologies are a tool. A tool that allows the development of better software. An extension of that, and the main topic of this thesis, is the tools that are used to support that development effort. It is often thought that good tooling and a good process lead to good software. This is a fallacy. Good people make good software, and the framework in they use to do so is provided by tools and methods.

The problem then becomes: to find the tools that fit the situation and the concluding remarks in chapter 5 provide some guidance for this choice. If we want to cut a tree in two we use a saw. In software development you will find a lot of people trying to cut that tree with a hammer. You might get there eventually, but the results won't be pretty and it will take a lot longer then it would when using the saw. The analogy is relevant to choosing a project planning and tracking tool for a project using an agile methodology. You want to use the right tool for the right situation. In some cases this will be pen and paper, in others it will be something extensive like VersionOne.

The challenge lies not so much in using a tool, but in choosing the right one. Unfortunately that is not a question that can be answered on the basis of this analysis. All the tools that were looked at have advantages and disadvantages, good points and bad ones. But an actual decision will have to be based on the situation the team that will use such a tool is in, and personal preference. An initial selection can of course be based upon the facts stated in this research, like whether a tool does or does not support release planning. But the main issue with tools, next to that they do what you want, is how they feel to the user when they are used. And this is so personal that the only way to find out is to try them out.

To conclude: Use the information in this article to help you decide what agile methodology best fits an organization and what tool has the features that best support that methodology

and the size of the projects that will be run with it. Let the team try it out and decide whether it is something they want to use. Then listen to them.

6.2 Additional research

This research thesis is not a true comparison of the various agile project planning and tracking tools. Its purpose was to find features in other tools that could point to good solutions for problems in PPTS. However during the writing of this report it became apparent that there is a lot of interest in the agile community for a true comparative review of the different agile planning tools currently available on the market. There are quite a few, many more then were reviewed in this report. So a further, in-depth, study of these tools and which corporate environments they best fit could be very interesting and socially relevant. Also a study of how the practices of each agile methodology map to the different tools could be investigated.

Secondly, there is very little data available on how projects run according to an agile methodology perform when compared to projects using other methodologies. This was a source of some frustration during the writing of this report. But it does provide an excellent opportunity for further study. There are a lot of projects being run in an agile context, especially in the USA and gathering data on them should not be very difficult. There is already a lot of data available on how projects perform with more established methods. So the main task is to gather the data from a significant number of agile projects and then to compare them to older projects using, for example, a waterfall type approach.

Appendix A

Compared tools

The following is a list of the tools that were included in the analysis, there are many more tools available but these were selected as those having the most to offer in general.

Open source tools

- Xplanner
  Location: http://www.xplanner.org
- XPWeb
  Location: http://xpweb.sourceforge.net

Commercial tools

- VersionOne
  Location: http://www.versionone.net
- ExtremePlanner
  Location: http://www.extremeplanner.com
- Rally
  Location: http://www.rallydev.com
- TargetProcess:Suite
  Location: http://www.targetprocess.com/

Appendix B

Tool comparison data set

The next pages contain the data set that was referred to in chapter 3. This data set was used in the comparison of most of the agile tools and contains data from a project that ran at Philips some time ago.

Work Breakdown Structure: wk21/22	(hours)	(hours)	(hours)	(hours)	
User Story Description	**Estimation**	**Total Initial Effort**	**Total Used**	**Total Last ToGo**	**CA***
1. Upload user photographs	2	2	2	0	Y
2. Graphical overview of user absents incl. print possibility via PDF file (2)	48	16	16	0	N
3. Quality Assurance wk 21/22			2	0	Y
4. Configuration Management wk 21/22			0	0	Y
5. Indirect hours wk 21/22			3.5	0	Y
6. Solve CR/PRs wk 21/22	18	18	3	0	Y
7. Create PDF for progress report	10	10	0.5	0	Y
8. Allow userstories to be prioritized	12	12	8	0	Y
9. Create a (graphical) resource allocation overview incl. Print possibility via PDF file with detailed project information		56	55.5	0	N
10. School tasks [Pascal]	16	16	30	0	Y
11. Miscelanious	0		9.5	0	Y
12. Modify Office Closed absent	6	12	11	0	Y
13. Releasing PPTS version 1.0	16	22	11	0	Y
14. Create a project allocation overview incl. print possibility via PDF file (1)		16	21.5	0	N
Total:	128	168	173.5		
1. Upload user photographs					
Implement uploading of photographs		2	2	0	
2. Graphical overview of user absents incl. print possibility via PDF file (2)					
Create graph		8	14	0	
Include holiday balance table on PDF file (on right side of legend)		6	1.5	0	
Remove second filter for holiday balance (use first filter instead)		2	0.5	0	
3. Quality Assurance wk 21/22					
PMP			0		
QA Check			2		
Reviews (preparation and meeting)			0		
4. Configuration Management wk 21/22					
Baseline Audit			0		
Build Fixing			0		
CCB Activities			0		
Release			0		
6. Solve CR/PRs wk 21/22					
PR #0001110: Tooltips are not displayed on correct position in progress report page.		2	2.5	0	
PR #0001111 Crash when saving togo hours on WBS page.		1	0	0	
PR #0001088 Burn down graph is not correct!		8	0	Cancelled	
PR #0001108 Entering tasks using 'Save and next' fails the second time.		2	0.5	0	
PR #0001101: Personal Report JavaScript code doesn't support negative and/or float values		1	0	0	
PR #0001127: Wrong burn down graph			0	0	
7. Create PDF for progress report					
Create PDF file for the report		8	0	0	
Review code and testcases		2	0.5	0	
8. Allow userstories to be prioritized					
Spike: Find out how to change order of user stories using java script.		4	8	0	
9. Create a (graphical) resource allocation overview incl. Print possibility via PDF file with detailed project information					
Create graph sceleton and legend		16	8	0	
Create algorithms for determining assigned hours/week		16	22	0	
Create PDF file for graph (incl. dividing it over multiple pages)		16	17.5	0	
View total absent days per week (as text overlay) incl. A check box to show or hide them.		8	8	0	
12. Modify Office Closed absent					
Also allow Holiday to be selected as Office Closed absent type.		2	3	0	
Verify operation of existing absent functions and correct them when necessary.		10	8	0	
13. Releasing PPTS version 1.0					
Converting existing production database to new database		12	7	0	
Transferring PPTS project from test database to new production database		2	4	0	
Check if all important buttons and menu items are protected by access control		4	0	0	
Make layout of all forms, reports and overviews etc. uniform.		4	0	0	

14. Create a project allocation overview incl. print possibility via PDF file (1)				
Create project allocation overview page and add menu entry		2	3.5	0
Create graph or table		6	9	0
Create algorithms for gathering data		4	8	0
Add project number to project properties		4	1	0

Work Breakdown Structure: wk23/24	(hours)	(hours)	(hours)	(hours)

User Story Description	Estimation	Total Initial Effort	Total Used	Total Last ToGo	CA*
1. Implement a maintenance mode for the website.	3	3	1	0	N
2. Improve burndown graph	9	9	0	0	Y
3. Create backlog page		22	10	0	N
4. Solve CR/PRs wk 23/24		15	5.5	0	N
5. Quality Assurance wk 23/24			0	0	N
6. Configuration Management wk 23/24			0	0	N
7. Indirect hours wk 23/24			1	0	N
8. Create public section of intranetsite for S2P2/EOG department		62	66	0	N
9. Add capabilities and courses		42	34	0	N
10. School tasks [Pascal] wk23/24	24	24	38	0	N
11. Refactor room reservation system	12	12	24	0	N
Total:	48	189	179.5		

1. Implement a maintenance mode for the website.				
Add menu items to enable/disable maintenance mode		0.5	0.5	0
Add maintenance mode variable in database		0.5	0	0
Prevent users to log in when maintenance mode is enabled		1	0.5	0
Add flag for each user which indicates if they can log in during maintenance mode		1	0	0

2. Improve burndown graph				
Add legend to burndown graph		3	0	0
Add line in burndown graph		6	0	0

3. Create backlog page				
Create and discuss proposal for new layout		4	4	0
Implement editing of user story priorities		10	4	0
Spike: Find out how to change order of user stories using java script.		8	2	0

4. Solve CR/PRs wk 23/24				
PR #0001135 Copying tasks works not well in all circumstances		5	4	Cancelled
PR #0001037 IterationPeriod should fall within ProjectPeriod.		4	0.5	0
PR #0001141 Allow only currently assigned developers be assigned to a task		2	0.5	0
PR #0001136 When 'copy task to userstory' is selected show iteration of US		4	0.5	Cancelled

5. Quality Assurance wk 23/24				
PMP			0	0
QA Check			0	0
Reviews (preparation and meeting)			0	0

6. Configuration Management wk 23/24				
Baseline Audit			0	0
Build Fixing			0	0
CCB Activities			0	0
Release			0	0

8. Create public section of intranetsite for S2P2/EOG department				
Create index page for public section and make this the default page		8	23	0
Add map of department and contact person info		8	2	0
Add people page		8	5.5	0
Create projects page		6	5.5	0
Refactoring of reused code		16	27	0
Create capabilities page incl. search function		16	3	0

9. Add capabilities and courses				
Design and create database tables		6	4	0
Create user capabilities and courses overview page		12	8.5	0
Create capability form		4	5.5	0
Create course form		4	4	0
Create search page for capabilities (detailed/non detailed)		16	12	0

11. Refactor room reservation system				
Refactor base class		4	10	Cancelled
Rebuild user interface		8	14	Cancelled

Work Breakdown Structure: wk25/26	(hours)	(hours)	(hours)	(hours)	
User Story Description	**Estimation**	**Total Initial Effort**	**Total Used**	**Total Last ToGo**	**CA***
1. Change Competences functionality	28	28	33.5	0	Y
2. Change projects overview (public web site)	20	20	8.25	0	Y
3. Show warning when over-assigning people (add/edit assignment) and show a visual indication on resource allocation graph	12	12	8.5	0	N
4. Quality Assurance wk25/26			0	0	N
5. Graphical overview of user competences (example S2P2)	36	36	27.25	0	N
6. Configuration Management wk25/26			0	0	N
7. Indirect hours wk25/26			18.25	0	N
8. School tasks [Pascal] wk25/26	0	8	8	0	Y
9. Refactoring wk25/26		8	8	0	Y
10. Create room reservation module based on module of old website		12	17	0	N
11. Solve CR/PRs wk25/26		8	12.5	0	Y
Total:	96	132	141.25		
1. Change Competences functionality					
Show description of level when adding/editing competences (form) and when viewing competences (overview).		4	2	0	
Create forms and handlers for add/edit/delete Categories, Competences and Levels		16	12	0	
Allow to add a competence when adding a course.		8	19.5	0	
2. Change projects overview (public web site)					
Allow HTML lay-out in customer project description and show this on the public website		4	0.5	0	
Change layout of projects overview on public website		12	5	0	
Add optional project property; a URL to the project website		3	2.25	0	
Add project property: Show/hide project on public site		1	0.5	0	
3. Show warning when over-assigning people (add/edit assignment) and show a visual indication on resource allocation graph					
Implement algorithm for calculating assignment percentage		8	4.5	0	
Show warning when over-assigning users while adding/editing assignements.		3	3	0	
4. Quality Assurance wk25/26					
PMP			0		
QA Check			0		
Reviews (preparation and meeting)			0		
5. Graphical overview of user competences (example S2P2)					
Create graph		32	23.25	0	
Create PDF file for the graph		4	4	0	
6. Configuration Management wk25/26					
Baseline Audit			0		
Build Fixing			0		
CCB Activities			0		
Release			0		
9. Refactoring wk25/26					
Create PDF files the same way everywhere.		8	8	0	
10. Create room reservation module based on module of old website					
Rebuild user interface		8	10	0	
Refactor base class		4	7	0	
11. Solve CR/PRs wk25/26					
PR #0001142 Velocity is taken from last iteration not from last completed iteration		4	2.5	0	
PR #0001161 After changing priority in BackLog, edit result in editing the wrong User Story		4	10	0	
Work Breakdown Structure: wk27/28	(hours)	(hours)	(hours)	(hours)	
User Story Description	**Estimation**	**Total Initial Effort**	**Total Used**	**Total Last ToGo**	**CA***
1. Quality Assurance wk27/28			0.75	0	N
2. Configuration Management wk27/28			8.5	0	N
3. Indirect hours wk27/28			6	0	N
4. Fix warnings issued when enabling error_reporting option in PHP	24	24	21.5	0	N
5. Allow to switch menu & styles to enable offline use of website.	68	68	59.5	0	N
6. Solve CR/PRs wk27/28		6	6	0	N
7. Fix warning when over-assigning people (add/edit assignment)	6	6	17	0	N
8. Create tool for easy updating of database	16	16	15	0	N

Total:	114	120	134.25	
1. Quality Assurance wk27/28				
PMP			0	
QA Check			0.75	
Reviews (preparation and meeting)			0	
2. Configuration Management wk27/28				
Baseline Audit			0	
Build Fixing			0	
CCB Activities			0	
Release			8.5	
4. Fix warnings issued when enabling error_reporting option in PHP				
Fix warnings		24	21.5	0
5. Allow to switch menu & styles to enable offline use of website.				
Make common CMenu class and specific derived classes		32	30	0
Make common CPageLayout class which handles page layout and use of style sheets.		32	27	0
Allow to swich between the menu/style sheets via constant in config file.		4	2.5	0
6. Solve CR/PRs wk27/28				
PR #0001163 Outlining of pictures and text is wrong		6	6	0
7. Fix warning when over-assigning people (add/edit assignment)				
Make the check which is performed working		6	17	0
8. Create tool for easy updating of database				
Create php page for updating the database		14	14	0
Create script for current updates		2	1	0

*CA = Customer accepted

Appendix C

Tool comparison results

The following pages contain a print out of the final excel pages with the results of the standard features for the comparison of the different agile tools.

(For Legend see bottom)	XpWeb	Extreme Planner	Rally	VersionOne	Xplanner	TargetProcess	PPTS
Basic Features							
Platform	Independent LAMP* solution	Windows NT/XP	Independent LAMP* solution, Hosted	Hosted or Local(Windows)	Web-Service	LAMP Hosted or Local	Independent LAMP* solution
Project							
Name	Yes	Yes	Yes	Yes	Yes	Yee	Yes
Description	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Other		User assignment	State, Owner, Notes	Statem, Owner, Sprint length, Dates	Hidden, Escape, Remind Developer to fill in hours, Wiki link	Start/End date	Number, Date of request, Customer, Contacts, Duration, Start/End date, Request handler, Preferred start/end date, Website link, commments
Actions	C,D,M	C,D,M	C,M	C,D,M	C,D,M	C,D,M	C,D,M
Release							
Name	x	Yes	Yes	Yes	x	Yes	x
Date	x	Yes	Yes	Yes	x	Yes	x
Other	x	Description	State, Version, Resources(hours),Notes	Description, Owner, Team, Reason	x	Goal, Create Iterations	x
Actions	x	C,D,M	C,D,M	C,D,M	x	C,D,M	x
Iteration							
Name	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Description	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Start/End date	Yes / Yes	Yes/Yes	Yes/Yes	Yes/Yes	Yes/Yes	Yes/Implicit	Yes/Yes
Real to Ideal time factor	Yes, LF	No	No	No	No	No	Yes,velocity
Other			Theme, State, Resources	Owner		Velocity, Duration, Release	
Actions	C,D,DaM,M	C,D,M	C,D,M	C,D,M	C,D,M	C,D,M,Mov	C,D,M
User Story							
Name	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Description	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Acceptance: Criteria / Flag	Textfield / No	No / No	No / Yes	Yes / Yes	No/Yes	No/No	No/Yes
Size estimate / Units	Yes / Weight(1-10)	Yes / IH	Yes / IH	Yes / General	Yes/Hours	Yes/Hours	Yes/IH
Priority setting (planning ?)	Yes	Yes	Yes	Yes	Yes	Yes	No
Risk	Yes	Yes	Yes	Yes	No	Yes	No
Other	x	Topic, Type, Value, Status, Release, Iteration, Requested by, Reference	State, Blocked, Date Accepted, Release, Iteration, Owner, Rank, Notes	Package, Status, Owner, Prod. Owner, Catergory, Reference, Release, Sprint, Team	Disposition, Customer, Tracker, Status	Effort, Initial estimate, Developers, Iteration, Release, Done	Iteration
Actions	C,D,DaM,M,Mov	C,D,M,Mov	C,D,M,Mov,S	C,D,M,Mov,S	C,D,M,Mov	C,D,M,Mov	C,D,M,Mov,Co
Task							
Name	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Description	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Owner	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Initial size (Unit)	ID	IH	Unspecified	Unspecified	IH	IH	IH
Done (Unit)	ID	IH	Unspecified	Unspecified	IH	IH	IH
Todo (Unit)	ID = Initial size - Done	IH = Estimated - Completed	Unspecified	Unspecified	IH	x	IH
Other	End/Start date, Pair programmer	Status	State, Blocked, Story Card, Notes	Category, Reference	Type, Disposition, Acceptor	x	
Actions	C,D,M,Mov	C,D,M	C,D,M,Mov	C,D,M,Mov	C,D,M,Mov,T	C,D,M	C,D,M,Mov,Co,Ca
Users							
Name	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Login Name	No	Yes	Yes	Yes	Yes	Yes	Yes
Password	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Email	No	Yes	Equals Login	Yes	Yes	Yes	Yes
Logged in Status	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Other	Description	Enabled	Display name, Session settings,	Phone, Role	Initial, Phone, Hide, System Admin		Acronym, Address, Phone, Date of Birth, Agency, Start/End date, Room, Courses, Competences
Actions	C,D,M	C,M	C,D,M	C,D,M	C,M	C,N	C,D,M
Access Control							
Role/Profile Based	Yes	Yes	Yes	Yes	Yes	No	Yes
Custom Definable	Yes	No	Yes	No,?	No	No	Yes
Assign user to projects	Yes	Yes	Yes	Yes	Yes	yes	Yes
Assign user to multiple projects	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Planning							
Release Planning	No	Yes	Yes	Yes	No	Yes	No
Date Driven Release Planning*		Yes	Yes	Yes		Yes	
Automatic planning		No	No	No		Yes	
What if analysis		No	No	No		No	
Manual		Yes	Yes	Yes		Yes	
Feature driven release planning*		No	No	No		No	
Iteration Planning							
Stories into Iterations	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Tasks into Stories	Yes	Yes	Yes	Yes	Yes	x	Yes
Velocity	No	No	No	No, is in graph	No	Yes	Yes
Backlog							
Implemented How	separate Unassigned stories list	Unassigned stories in story list	Unscheduled story card list	Backlog	No	Story list	Backlog
Sorted by	broken	All	All	All	x	All	x
Planning Metrics							
Velocity	Yes	No	No	No	No	Yes	Yes
History based?	Yes	No	No	No	No	No	Yes

Units	SP, ID, RD	n/a	n/a	n/a	n/a	IH	percenatage, 1/LF

What	Total effort	Estimated efficiency	Resources	Total effort (Backlog)	Total estimate	Assigned/possible effort per developer	Total estimate
Units	ID	Estimated / Actual Hours	ID	Unspecified	IH	IH	IH
Visualizations	Numerical	Bar chart	Numerical	Numerical	Numerical	Numerical	Numerical

What		Estimated hours	Planned estimates	Total effort (Tasks)		Planned/Available hours in iteration	
Units		IH	ID	Unspecified		IH	
Visualizations		Numerical	Numerical	Numerical		Numerical	

What			Available				
Units			ID, Resources - estimates				
Visualizations			Numerical				

Tracking	How does the tool support the tracking !!						
Release							
Numeric		No	Yes	Yes	No	Yes	No
Visual		Yes	Yes	Yes	No	Yes	No
Iteration							
Numeric		Yes	Yes	Yes	Yes	Yes	Yes
Visual		Yes	Yes	Yes	Yes	Yes	Yes

Tracking Metrics							
Release							
What		Open Stories / Iteration	Accepted and Scheduled *1	Backlog, Done vs Todo		% Done vs Time elapsed *1	
Units		Stories	Story Cards	Estimates, stories		percentage	
Representation		Burn Down	Numerical	Bar Chart		Numerical	

What		Finished Stories	Passing and Total *2	Backlog, Done vs Todo		Release status *2	
Units		Stories	Test cases	Estimates, Tasks		Good, Normal, Bad	
Representation		Numerical	Numerical	Bar Chart		Color	

What			Active and Total *3	Member Workload		Stories Done *3	
Units			Defects	Task estimates		Stories	
Representation			Numerical	Horizontal Bars		Numerical	

What			Release Cumulative Flow Chart	Velocity		Open / Resolved bugs *4	
Units			Points (general unit)	Unspecified		Bugs	
Representation			Bar chart	Bar chart		Numerical	

What			Release Burn Down Chart			Test cases passed	
Units			Points (general unit)			Pass/fails	
Representation			Bar chart			Bar chart	

Iteration							
What	Total Completion	Total Completion	*1, *2, *3 As above	Backlog, Done vs Todo	Hours, Estimated / Actual / Reamining	1,2,3,4 as above for iterations	Iteration Burn Down Chart
Units	SP,IH, ST, Percentage	ST, Ta	Same as above	Estimates, stories	IH	as in Release	Hours
Representation	Numerical, Progress Bar, Num, Num	Numerical	Numbers	Bar Chart	Numerical	as in Release	Bar chart

What	Total time needed	Status of tasks: todo, active, completed	Iteration Cumulative Flow Chart	Backlog, Done vs Todo	Iteration Progress	Iteration Burn Down Chart	Iteration Burn Down Data
Units	ID	Ta	Points (general unit)	Estimates, tasks	Estimated / Actual Hours	Hours	Hours
Representation	Numerical	Graphical overview	Bar chart	Bar Chart	Chart	Bar chart	Numerical

What			Iteration Burn Down Chart	Estimation Change	Iteration Burn Down Chart	Bugs progress chart	
Units			Points (general unit)	story estimate	Hours	Open/Closed bugs per iteration	
Representation			Bar chart	Numerical	Bar chart	line chart	

What				Daily Burn down	Completed estimated Hours	Velocity	
Units				Task estimates	IH	Hours	
Representation				Burn-down, line based	Pie-chart	Line chart	

What				Task Trend lines	Completed actual Hours	Estimation accuracy	
Units				Done / Todo, This / Prev sprint	IH	%	
Representation				Line chart	Pie-chart	Line chart	

What				Daily Backlog Trend		Test cases passed	
Units				Unspcified, This / Last Iteration		Pass/fails	
Representation				Line chart		Bar chart	

Story							
What	Total Completion	Total Completion	Status	Status	Progress	Progress	Progress
Units	Percentage	Ta, Percentage	Backlog, Defined, In-Progress, Completed, Accepted, Blocked in any of the previous 5 phases	Planned, In progress, Completed	IH	%	%
Representation	Progress Bar	Numerical, Progress bar	Graphical Icons	Select box	Progress Bar	Numerical	Progress bar

Task							
What	Total Completion	Total Completion	Status		Progress		Progress
Units	Percentage	IH,Percentage	Defined, In-Progress, Completed or Blocked		IH		%
Representation	Progress Bar	Numerical, Progress bar	Graphical Icons		Progress Bar		Progress Bar

Usability / User Interface							
Interaction							

Multiple item move/delete	No	Yes on delete, No on move	Yes, Move	No	No	No	Yes, move and copy
Drag and Drop	No	No	Yes	Yes	No	No	No
Tooltips	No	No	Yes	Yes, in report section	No	No	No
Popup Calender data selection	Broken	Yes	Yes	Yes	Yes	Yes	Yes
Intuitive Interface	o	o	o	+	o	o	o

Tool							
Responsiveness	+	++	o	+	++	o	o
Ease of use	o	+	+	+	+	+	-
Installation	o	+	++	++	--	+	o
Upgrading	-	o	++	++	x	x	o
Sorting	+	+	++	++	+	+	-

Overviews							
Project overview	+, Pr	+', Pr, It, Ta	-', unclear	+', Pr, Re, It	-', It	o, St	o, Pr
Release overview	n/a	-	+', graphical timeline	+	x	-', Re	n/a
Iteration overview	o, It, St, Ta	Yes, Graphical over view of one iteration showing all stories and tasks and their status. ++***	o, St, Ta	+', St	+', It	o, It	"+ , St, Ta"
Stories overview	o	+	o	+', St, Ta (collapsible)	+", St'	+'", St", Ta	o
Tasks overview	o	+	+	+	+	x	o

Extra features							
General							
Code Version Control System	Yes, linked	No	No	No	No	No	No
Bug Tracker	No	No	Yes, Integrated	No	No	Yes	No
User Documentation	Docs, Javadocs, Design docs, Unit Tests, all Linked.	Docs can be linked to stories	Yes, Attachments	No	No	Yes	No
Project Website	Yes, linked	No	No	No	Yes, Wiki link	No	Yes
Additional	Metaphors list	History of all user actions	See Report	History	Integration, History	Blog, Notes, Files	Absents, Room reservation, Workloads, Bugets, Risks, Competences
Import/export excel	No	Yes	Yes, XML too	Yes	Yes	Yes	No
Help / Documentation							
Help Documentation	Yes, Web-based, integrated, Very Good***	Yes, Link to basic user manual	Yes, User manual, tutorials, tooltips and in-site	Yes, User Manual	Yes, Documentation	Yes	No
Installation Docs	Yes, on website	Yes, on website	Yes	Yes	Yes	Yes	Yes

Notes:	Lots of bugs, sorting doesn't work and neither does moving stories when you delete them	No bugs seen	No bugs seen	No bugs seen	No bugs seen	No bugs seen	
	Has a calender which shows all tasks, crowded picture and seems te be not very usefull.	Release feature seems useless except that it exists	Overviews can be a bit annoying at times, because there are a lot of them. Also a lot of numbers and not that many bars and completion bars which are easier to understand		Simple program, easy to use except ofr adding users to projects which seems either broken or strangely implemented.	Mid weight program, easy to use but layout of menu's and items could be better and needs better overviews	
	Numbers on iterations, stories and tasks is not consistant when you delete items		Program has a more rigid approach to the development process with a definite and specific workflow.				
	Calculated end data is not consistent when you change the start date, leads to eroneous behaviour.		Has extra means of describing content, such as requirements, features, non-functional requirements, use cases, defects and defect tracking.				

*Topic = General category of the story for search purposes.	*LAMP = set of free software programs used to run dynamic websites. Usually refers to Linux, Apache, Mysql, Perl / PhP / Python.	*Date driven release planning = Release is a set timebox, e.a. hard deadline, so amount of functionality in a release can shift but not the date	*Feature driven release planning = Release is defined as a set of functionality, this set if fixed so if stuff takes longer the release date slips.

*EOI = End of iteration	Rr = Project	Scale	ACTIONS	
*SP = Story Points	Re = Release			
*ID = Ideal Days	It = Iteration	Very Good = ++	CREATE	C
*IH = Ideal Hours	St = Story	Good = +	DELETE	D
*RD = Real (actual) Days	Ta = Task	Neutral = o	DELETE AND MOVE (move to somewhere when deleted)	DaM
*RH = Real (actual) Hours		Bad = -	MODIFY	M
/* = interesting feature		Terrible = --	MOVE	Mov
			COPY	Co
			MERGE	Mer
			SPLIT	S
			Set Active	SA
			Set Hidden	
			Cancel	Ca
			T	Set Timed

Appendix D

Compared tools extra features summary

The next page contains a short summary of the extra features found in the compared tools which could be of use in deciding what must be changed in PPTS. This is an extract of the Results in chapter 5.

Tool :	XpWeb	Extreme Planner	Rally	VersionOne	Xplanner	TargetProcess
Good Features						
	LAMP solution so will run on most operating systems.	Has a windows installer, easy to use but not multi-platform.	Useful for managing larger projects	Clearer then Rally due to less pages and clearer menu interface	Simple tool with limited functionalilty.	Clean interface, but does not have the most logical setup.
	Excellent help functionality	Releases supported although in a minimal fashion.	Integrated defect tracking and test cases.	Good intuitive project setup guide worked into the meny structure.	Easy to use due to it's limited scale and a small learning curve	Allows developers to blog messages, these are shown in a bar on the left on most pages.
	Automated database initialization from the web interface of XPWeb	Import and Export stories to and from Microsoft Excel	General search function for locating stories, defects or tasks in the project. Very usefull in large projects with hundreds of stories.	Can be run hosted or local as a lamp solution, so cross platform and ability to choose depending on the situation.	Ability to add notes to most of the screens for inter team or other communication	Also allows the posting of files on a kind of bullitin board.
		Extensive sorting functionality including custom default sorting method.	Helpful movies and guides to help in getting started with the tool. Good since the interface is not very intuitive.	Clear and intuitive user interface for the most part.	Has warnining features for people who did not fill in their hours yet by the end of the day.	Very good resource management including work load per team member.
		Clean interface.	Easy planning of Releases and Iterations via drag and drop functionality and the options to move multiple stories or tasks.	Lots of metrics both numerical and grpahs that show how the projects is doing in time.	Simple and clear project hierarchy	Auto-iteration planning feature.
		Excellent iteration overview page, though it becomes cluttered with a lot of stories in an iteration.	Very extensive tracking features both numerical and graphical for both iterations and releases.	Useful for large projects but van also be used for smaller ones.	Good metrcis for the scope of the tool with a special focus on pair programming.	Indicator of planned effort in an iteration against available resources.
		Simple project status overview page when the user logs in.	First tool that has indicators which show how much space remains in an iteration while planning.			indicator of effort done against time elapsed in an iteration.
		Brun-down graph to visualize iteration status.	Tracking of individual effort and assigned tasks via a sort of home page per user.			
			Time-line chart for the entire project gives a nice overview of all planned activities.			
			Actions and their feedback is usually on the same page.			
Bad Features						
	Lots of bugs, sorting and calendar seem broken in t his version.	Release functionality included is very limited, namely to a release date and adding stories. But there are no usefull metrics and tracking tools for releases.	Rally's internal structure for representing projects is quite complex and as such makes working with the tool more difficult. This also results in a longer learnining curve.	Charts and reports are linked to their subject and not all viewable from one page, not that bad but can be annoying.	Extremely unfriendly to install, should only be done by an IT professional.	Automatic iteration planning is not very consistent with team empowerment and agile methodologies in general.
	No release planning.	No visual or other help in plannig a iteration or release, such as resources used.	Interface is not intuitive in planning a projects, although the tool does allow for great up-front planning.	Not too much feedback in the planning cycle.	No release support, should thus not be used for overly large projects	Interface is not very intuitively strcutured.
	Too many actions involved in basic tasks like booking time on tasks.	A number of editing options are counter intuitively placed and as such misleading.	Releases and Iterations are planned using different entities, this is needlessly confusing,		Very little feedback on planning.	Users can set velocity but this does not seem to be dynamic or history based after it is set.
		Choice of starting a task when a developer is assigned instead of when actual hours are booked is debatable	Tracking information is ditributed over a number of pages without one central overview.			User stories list page takes way too much space which leads to a lot of scrolling and a bad overview.
		Iteration overview gets cluttered easily, needs to be made more compact.				Metrcis are spread out over the tool without a central overview.
						Cumbersome manner of filling in time spent on tasks.

Bibliography

[ABB+98] A. Anderson, R. Beattie, K. Beck, D. Bryant, M. DeArment, M. Fowler, M. Franczak, R. Garzaniti, D. Gore, B. Hacker, C. Hendrickson, R. Jeffries, Doug Joppie, D. Kim, P. Kowalsky, D. Mueller, T. Murasky, R. Nutter, A. Pantea, and D. Thomas, *Chrysler goes to extremes*, Distributed Computing **1** (1998), no. 10, 25–28.

[ASRW02] P. Abrahamsson, O. Salo, J. Ronkainen, and J Warsta, *Agile software development methods, review and anlysis*, Online, 2002, Available at : http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf.

[Bec00] Kent Beck, *Extreme programming explained, embrace change*, Addison-Wesley, 2000.

[Bro87] F. Brooks, *No silver bullet: Essence and accidents of software engineering*, Computer (1987), 10–19.

[CF01] Ward Cunningham and Martin Fowler, *The agile manifesto*, Online, 2001, Available at : http://www.agilemanifesto.org/.

[Coc04] Alistair Cockburn, *Crystal clear : A human-powered methodology for small teams*, Addison-Wesley, 2004.

[Coh05] Mike Cohn, *Agile estimating and planning*, Prentice Hall PTR, 2005.

[Cus98] Michael A. Cusumano, *Microsoft secrets: How the world's most powerful software company creates technology, shapes markets and manages people*, Free Press, 1998.

[Hig00] J. Highsmith, *Adaptive software developmenadaptive software development - a collaborative approach to managing complex systems*, Dorste House Publishing, 2000.

[LB03] Craig Larman and Victor R. Basili, *Iterative and incremental development: A brief history*, Computer **36** (2003).

[Met] Advanced Development Methods, *Controlled-chaos software development*, Online, Available at : http://www.controlchaos.com/download/Controlled-Chaosnt.pdf.

[PF02] S.R. Palmer and J.M. Felsing, *A practical guide to feature-driven development*, Prentice Hall, 2002.

[SB02] K. Schwaber and Beedle, *Agile software development with scrum*, Prentice Hall, 2002.

[Sch96] Ken Schwaber, *Conctrolled choas: Living on the edge*, Cutter IT Journal **9** (1996), no. 3.

[Sta97] J. Stapleton, *Dsdm: Dynamic systems development method, the method in practice*, Addison-Wesley, 1997.

[Sut04] Dr Jeff Sutherland, *Agile development: Lessons learned from the first scrum*, Cutter Agile Project Management Advisory Service, Executive Update **5** (2004), no. 20.

[Sut05] Jeff Sutherland, *The roots of scrum: How japanese lean manufacturing changed global software development practices*, Online, 2005, Available at : http://jeffsutherland.com/scrum/RootsofScrumJAOO28Sep2005.pdf.

[WU01] Laurie Williams and Richard L. Upchurch, *In support of student pair-programming*, ACM SIGCSE Bulletin **33** (2001).