NTNU – Trondheim
Norwegian University of
Science and Technology

Promoting reflection in agile software development teams using GitHub data

Marius Nedal Glittum
Even Stene

Master of Science in Informatics
Submission date: May 2013
Supervisor: Monica Divitini, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Preface

This is the Master's thesis of IT3900, written by Even Stene and Marius Nedal Glittum from August 2012 to May 2013 at the Norwegian University of Science and Technology(NTNU).

We would like to thank our supervisor, Monica Divitini, for her valuable guidance while writing the thesis. We would also like to thank all who participated in the different evaluations, your comments and feedback was greatly appreciated.

Marius Nedal Glittum

Even Stene

Abstract

Agile software development teams work with several different artifacts on a daily basis, and by interacting with these artifacts users are involved in work related experiences. By revisiting these experiences and reflecting upon them, users can evaluate and improve how they solve everyday working tasks. Boud et.al defines reflection as a process where the experience is revisited, feelings are re-attended and the experience is re-evaluated[Boud et al., 1985]. Furthermore work and reflection on work are shown to be strongly connected[Schön, 1983][Chaiklin and Lave, 1993]. Reflecting on work experiences give a better understanding of the experience itself, allowing for conclusions and lessons learned to be made. Reflection transforms experience into knowledge which can be applied to solve challenges in the everyday working environment.

The main focus of this thesis was to develop a technological tool to collect project artifacts and connect these to work experiences, in order to enhance reflection both individually and collaboratively in agile software development teams. The tool was developed using a daily delivery cycle. Design choices were made on the basis of available theory, literature and related tools concerning reflection and agile development. Three evaluations were conducted; A usability study, an expert review with an expert in the field of agile software development and a focus group evaluation consisting of eight software developers working in an agile team.

The work conducted resulted in a Grails web-application, where users connect their daily experiences with project artifacts collected from a Version-control system. These daily reflection notes can be used both individually and collaboratively in a team as preparation for agile retrospective sessions. The tool continuously collects work-related project artifacts and presents these in order for users to *revisit* their work that day. The application aims to trigger reflection on user experiences and storing the outcome in notes for later use and sharing.

This thesis, the developed tool and its evaluation contributes with an increased understanding of how reflection in agile software development teams can be improved, by connecting experiences with work related project artifacts.

Sammendrag

Smidig programvareutviklings team jobber med flere forskjellige gjenstander på en daglig basis, og ved å samhandle med disse gjenstandene brukerne er involvert i arbeidsrelaterte erfaringer. Ved senere å gjennomgå disse erfaringene og reflektere over dem, kan brukerne evaluere og forbedre hvordan de løser dagligdagse arbeidsoppgaver. Boud et.al definerer refleksjon som en prosess der opplevelsen er gjenskapt, følelser gjenskapes og opplevelsen er re-evaluert [Boud et al., 1985]. Videre arbeid og refleksjon over arbeidet er vist å være sterkt forbundet [Schön, 1983; Chaiklin and Lave, 1993]. Reflektere over arbeidserfaringer gir en bedre forståelse av selve opplevelsen, slik at konklusjoner og lærdommer kan trekkes. Refleksjon transformerer erfaring til kunnskap som kan brukes til å løse utfordringer i et hverdagslig arbeidsmiljø.

Hovedfokus for denne avhandlingen var å utvikle et teknologisk verktøy for å samle prosjektet gjenstander og koble disse til å jobbe erfaringer, for å styrke refleksjon både individuelt og i samarbeid i smidige programvareutviklings team. Verktøyet ble utviklet ved hjelp av en daglig leverings syklus. Design valg ble gjort på bakgrunn av tilgjengelig teori, litteratur og relaterte verktøy om refleksjon og smidig utvikling. Tre evalueringer ble gjennomført; En brukbarhets studie, en ekspert gjennomgang med en ekspert innen smidig soft-ware utvikling og en fokusgruppe evaluering som består av åtte programvareutviklere som arbeider i en smidig team.

Arbeidet resulterte i en Grails web-applikasjon, der brukerne koble sine daglige erfaringer med prosjektet gjenstander samlet inn fra et system for versjonskontroll. Disse daglige refleksjonsnotatene kan brukes både individuelt og i samarbeid i et team som forberedelse for smidige retrospektive økter. Verktøyet samler kontinuerlig arbeidsrelaterte prosjekt gjenstander og presenterer disse for brukerne for å revidere sitt arbeid den dagen. Applikasjonen har som mål å utløse refleksjon på brukeropplevelser og lagring av utfallet i notater for senere bruk og deling.

Denne avhandlingen, prototypen og evalueringen bidrar med en økt forståelse av hvordan refleksjon i smidige programvareutviklings team kan forbedres, ved å koble erfaringer med arbeidsrelatert prosjekt gjenstander.

Contents

List of Figures

Chapter 1

Introduction

This thesis presents a design-science project where the goal was to design a web-application tool as a support for learning and reflection in agile project development teams. The tool was given the name *PeacefulBanana* and the name has since remained unchanged. The purpose of the tool is to make collection of experiences easier during a project development process, and enhance learning outcomes from reflection sessions.

The tool collects data from the development process and presents them to the users in a tailored way. Users can annotate these data with their own experiences, including connecting feelings and emotions to these experiences. The application presents these data and experiences for use in both individual and collaborative reflection settings. The challenge will be to collect, annotate and present the data in a way that helps the user understand what they have experienced and learn from these experiences. That is helping them reflect on the experiences made and learn from them. Boud et.al[Boud et al., 1985] presents that;

> *Reflection is a process where the experience is revisited, feelings are re-attended and the experience is re-evaluated*

This means that it is important to tailor the reflection process in order to support and achieve learning from experiences. Using technology to collect data and experiences from everyday work, and the potential for using this to support reflection has been shown to be growing [Li et al., 2011]. In this thesis we utilize technology for promoting reflection, by trying to capture everyday data related to work experiences, make them available for reflection and store them for later use. The quality of data collected will have an impact on how easy it is for users to revisit the experience and reflect upon it. By using technology we hope to make the challenge of capturing the experience with it's ideas and feelings to support the reflective process easier to overcome.

1.1 Context & Domain

For many years now, agile methods in software development have been widely used in software development teams. Agile methodologies are software development methods that is based on iterative and incremental development. This means that requirements and solutions are dynamic and comes as a result of collaboration between self-organizing teams. Agile methods focuses on adaptive planning, iterative development and encourages teams to respond to changes in a flexible way. Agile development has been reported to have major improvements over more traditional development methods [Dybå and Dingsøyr, 2008].

The agile manifesto presents a principle which states that an agile team should regularly reflect on how to become more effective, and tune its behavior accordingly[Beck et al., 2001]. Continuously improving through introspection is a vital part of agile methods and is applied i.e. retrospectives[Beck, 1999; Derby and Larsen, 2006; Maham, 2008]. Cockburn impose that a vital part of agile practice should be conducting regular retrospective workshops aimed at reflection and process tune-up[Cockburn, 2006]. Agile methods focus on continuous iterations repeating the same development steps, and thus progressing. Retrospectives in agile development processes are most often performed after each iteration. This is done by gathering the team and reflecting on their way of working, so that improvements for the next iteration can be identified [Derby and Larsen, 2006; Drury et al., 2011]. This enables agile self-governing teams to react quickly to changes, and make modifications accordingly[Drury et al., 2011]. Retrospectives could also support communication and interaction within the team, which is important for agile development.

There are however shortcomings of applying retrospectives in agile teams. Three recurring challenges for agile teams have been discovered in [Gulliksen Stray et al., 2011]. Findings from the authors show that developers often solve tasks according to what they find interesting, instead of solving tasks with the highest priority which is a practice that does not align with the agile focus of delivering the highest prioritized functionality. This also leads to a low competence-overlap within the team. The authors also discuss the challenge of communication problems within the team, concerning how critical decisions are taken by project management without involving the rest of the team. Though often, team-members are unfocused and do not involve themselves during meetings. Another challenge identified for teams, is the missing ability to turn results from a retrospective analysis into applicable knowledge, improving everyday tasks.

Agile teams tend to focus primarily on short-term issues that were identified for a single iteration, and not on long-term strategic issues[Drury et al., 2011]. Reflection on work related experiences enables users to reflect and derive conclusions from the reflection [Korthagen and Vasalos, 2005]. This turns experiences into knowledge that can be applied to the everyday challenges of work and also creates an individual aspect, where the reflection on previous work can be applied to future work challenges. In addition to this individual aspect, reflection has been shown to have strong social aspects, and is often accomplished collaboratively between users in such agile teams [Høyrup, 2004]. Therefore a challenge will be to easily allow users to identify their common tasks and shared work experiences. In software development projects in the industry, it is often a challenge for teams to prioritize retrospectives, as there is other work that is seen as more important [Kasi et al., 2008]. For students, revisiting experiences and reflecting on these is often seen as unnecessary and in the way of other tasks, i.e. writing code, testing or documentation. Another challenge is collecting data of sufficient quality to support this reflection, and also how to share this data, experiences and reflection with the rest of the team.

In this thesis we developed a tool in order to address the challenges of collecting data of sufficient quality to support reflection and how to share these data and experiences in order for users to revisit and reflect upon them. We wish to allow teams to not only reflect on experiences for the last iteration only, but also allow users to go back in time several iterations, in order to identify long-term issues and trends within the team. The tool is aimed at supporting reflection by scaffolding the collection of relevant project artifacts, and prompting users to annotate these artifacts with their experiences and feelings. The tool is designed and developed for software development projects using an agile development process, and will be evaluated with a usability study, a focus group consisting of software developers and also an expert in the field of agile methodologies.

1.1.1 Core Concepts

In this section we will briefly introduce some concepts which the application developed for this thesis builds upon.

Learning from experience

According to David Kolb[Kolb D.A, 1975] for a learning experience to occur there must exists certain abilities in the learner. First the learner must be willing to actively be involved in the experience Secondly the learner must be able to reflect on the experience and third, the learner must possess and be able to use analytical skills to conceptualize the experience. Finally the learner must have skills for decision making and problem solving to be able to create new knowledge outcomes based on the experience.

Software development

Software developers generate a lot of data when developing software, when committing data to version control systems or closing tasks and issues in a project management tool. During an agile development cycle, tasks is distributed among the developers and they make decisions either individually or collaboratively. Normally this data is never revisited, but they contain vital information on choices the developers make daily. It is important to make collection of information a part of developers normal day-to-day activities , as can be based on the model from [Krogstie and Divitini, 2009].

Version-control systems

Version-control systems is the management of changes to documents, source files or other collections of information. These artifacts are usually identified by a 'revision'[1], when creating a revision a lot of data is stored together with the revisioned files. I.e. who committed the data, when it was committed and what files where affected by the change.

In software development, version-control systems can be used for documentation and configuration of a wide variety of files as well as source code. As teams develop software, it is common for developers to create and work on

[1]An unique identifier normally a number or string.

different versions of a system at the same time. Different versions can be tagged in the version-control system, so it is easy to go back in time to a specific state, i.e. looking up the tag for when the last stable version was created.

1.2 Research Question

In this section we will describe the research questions for this master thesis:

Main RQ

- How to promote experienced-based reflection based on project artifacts collected from version-control systems?

Sub RQ1

- How to scaffold collection of data in order to promote reflection?

Sub RQ2

- How to increase the tendency to reflect on experiences, both individually and as a team?

Sub RQ3

- How to bring together contributions from multiple users, and sharing these in a collaborative environment?

1.3 Research Method

Design as an artifact

By definition, the result of design-science research in Information Science is:

> *A purposeful IT artifact created to address an important organizational problem. It must be described effectively, enabling its implementation and application in an appropriate domain.* [Esearch et al., 2004]

Markus et al.[Markus et al., 2002] identified that a developed artifact is only significant by looking at some significant questions:

- Can the artifact be constructed?
- Can the artifact perform appropriately?
- Is the result important to the information science community?

We will create a proof of concept prototype tool for promoting reflection on work experiences experienced, by utilizing project artifacts collected from version-control systems. Our goal was to create a tool that can discover new capabilities in the domain of reflection on experiences and learning from these, as well as support the existing capabilities in an efficient way. Evaluating the tool in *real use* situations is necessary in order to discover if the artifact can enhance the reflection process as it is today.

Research Guidelines

The *Design-Science Research Guidelines* presented in [Esearch et al., 2004] will serve as the basis of this design-science research. The guidelines were created in order to assist researchers to understand the necessary requirements for effective design-science research.

1.3.1 Design as a research process

Development of the application was done in development cycles, inspired be the regulative cycle presented by Wieringa [Wieringa, 2009], and is shown in Figure 1.1. The development process and the regulative cycle we adopted for developing the application is further detailed in Section 1.3.2 and Section

1.3.3. The early parts of the development process was used to develop ideas and basic mock-ups of the application and its design. The initial prototype design was based heavily on the first concept and mock-ups created, in addition to recurring feedback from users and our supervisor. Early parts of development resulted in a prototype with functionality for individual reflection use. The next goal was to incorporate teams and support collaborative reflection in the team, building and improving on the basic functionality present in the tool. In the later parts of the development, the focus was the integration of individual reflection notes, into the team collaborative areas. This resulted in a tool for both individual and collaborative aspects for reflection and learning from experiences. We used the tool ourselves during development, as it would be used in a real working environment. In this way we ensure that all functionality is as expected, and also identify limitations early in the design.

Finally we performed three different evaluations. First we conducted a usability test with computer-science students, and also an expert review with an expert in the field of agile software development. The last evaluation we conducted was a focus group evaluation with an agile software development team using GitHub as their main version-control system. Evaluating these separately provided three sets of feedback which we could compare in order to possibly see if any patterns emerged.

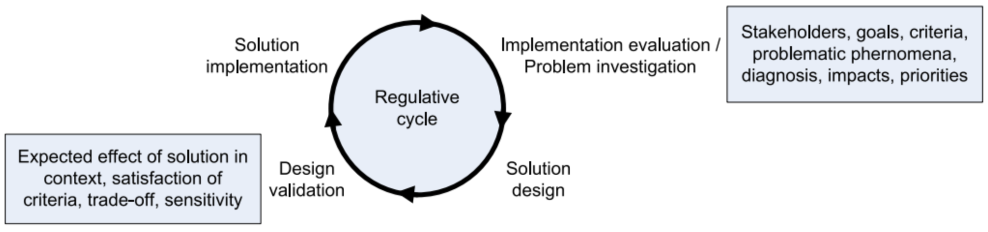

Solution implementation
Implementation evaluation / Problem investigation
Stakeholders, goals, criteria, problematic phernomena, diagnosis, impacts, priorities
Regulative cycle
Expected effect of solution in context, satisfaction of criteria, trade-off, sensitivity
Design validation
Solution design

Figure 1.1: Regulative cycle development in Design-Science research [Wieringa, 2009]

1.3.2 Development process

Figure 1.2 shows the research model used for the development of our prototype from initial concept and ideas, to the final implementation and evaluation. The model depicts how we started the process by reading theory(top part of model) in order to get a better understanding of the core concepts of reflection on experiences(See Chapter 3 for the thesis background).

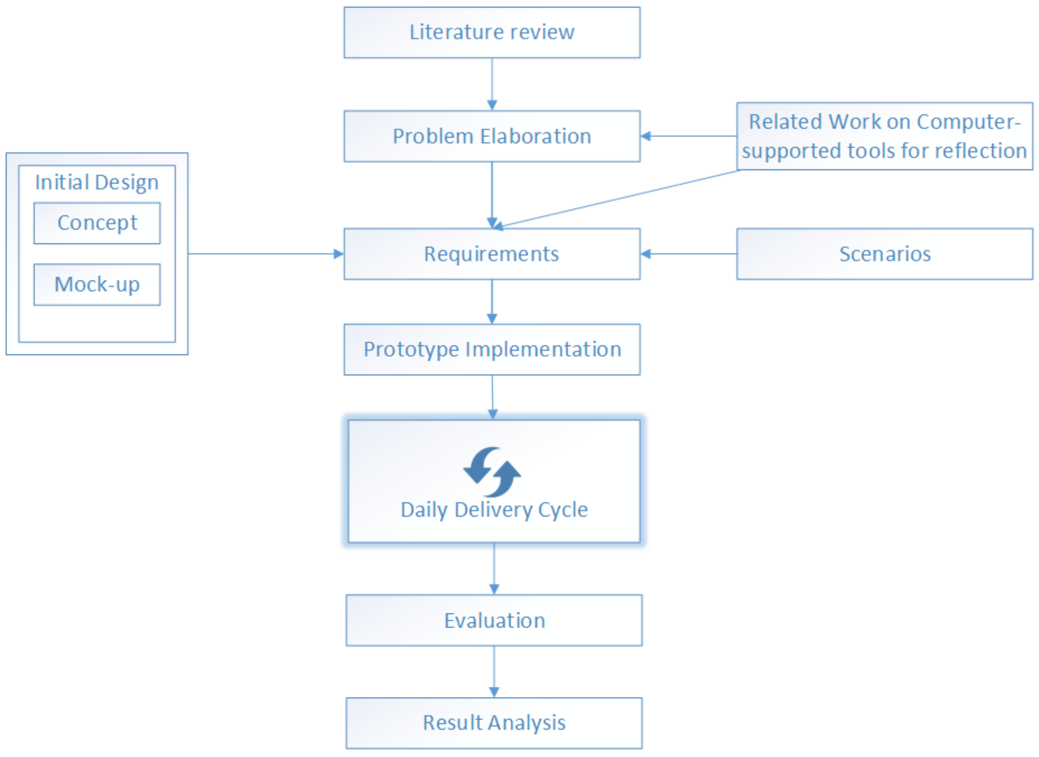


Figure 1.2: The PeacefulBanana research model

We conducted a literature review (Section 4.1) and looked at related work done on computer-supported tools for reflection and GitHub related tools (Section 4.2). This gave us a basis on which to build a problem elaboration on(See Chapter 5) and this stage can be seen as the second from the top in Figure 1.2. The problem elaboration with our application scenarios, in addition to related tools, led us in turn to the process of creating the requirements for the application (See Chapter 6). The initial design stage to the left in the model, with the application concept and mock-ups also helped further identify requirements for the application, together with feedback from our supervisor.

Based on the stages from literature review to requirements in the model shown in Figure 1.2, we started implementing the PeacefulBanana application. These implementations were conducted using a daily delivery cycle, which we used throughout the development phase of the application. This cycle is further described in Section 1.3.3.

The last two stages of our research model is Evaluation and Result analysis. Evaluation consisted of a usability study, an expert review and a focus group (See Chapter 9). Based on the findings from these evaluations we were able to perform a result analysis.

1.3.3 Daily Delivery Cycle

Figure1.3 shows an illustration of the daily delivery cycle used throughout development. This daily delivery cycle was inspired by the regulative cycle presented by Wieringa[Wieringa, 2009], shown in Figure 1.1. How this cycle fits in the overall stages of research is shown in Figure 1.2.

The cycle starts with a set of requirements. These requirements are initially based on user-stories and scenarios. In subsequent iterations of the cycle, the requirements were then updated based on feedback from from experts and users of the application. A task list is then created from the set of requirements, where the highest prioritized requirements was implemented first. This task list acts as a backlog [2] for the application. On GitHub we created the more general tasks as *Milestones* and the more specific tasks as *Issues* connected to these milestones[3].

[2]The backlog is the list of work the developers must address during the current iteration
[3]GitHub Issues & Milestones: `https://github.com/features/projects/issues`

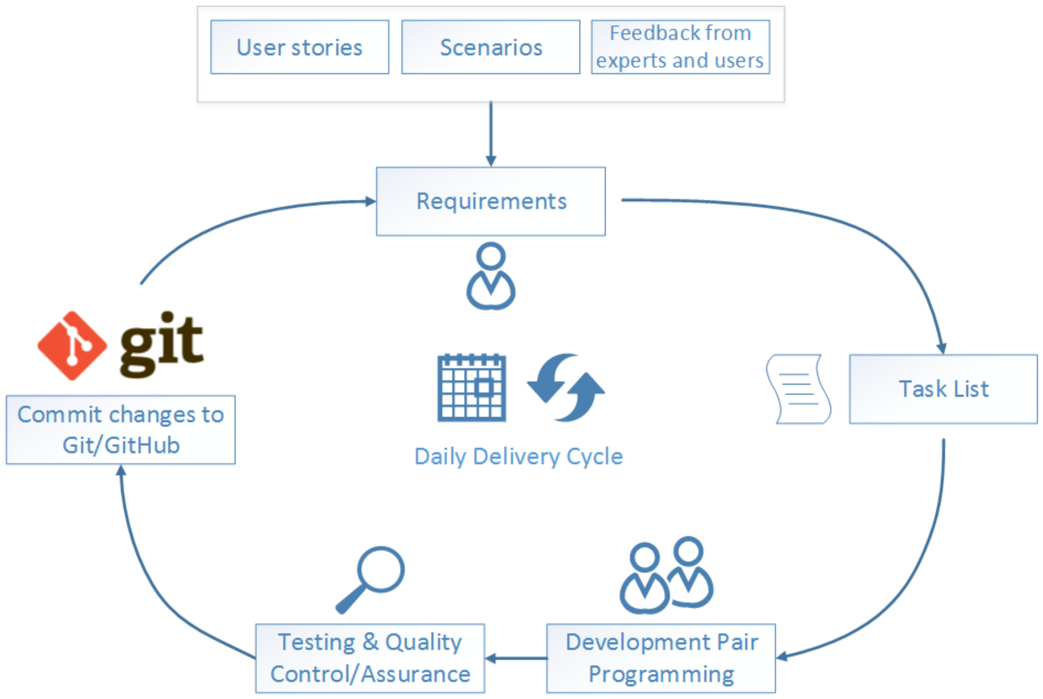

User stories
Scenarios
Feedback from experts and users
Requirements
git
Commit changes to Git/GitHub
Daily Delivery Cycle
Task List
Testing & Quality Control/Assurance
Development Pair Programming

Figure 1.3: The PeacefulBanana daily delivery cycle

Development was conducted primarily using pair-programming[4], so that we could review each line of code as it was written. Whenever a task implementation was completed, it was tested manually in the application, as well as by automatic tests. When a feature was accepted and working as intended, we submitted the code to our project repository on GitHub. This ensured an iterative approach to development, expanding our application with more and more quality-assured functionality.

Dogfooding

While developing the application, we continuously used the application ourselves, this is a concept called dogfooding.

Dogfooding is a concept where you 'eat your own dog food', this means that the developers are testing their own product while developing it[Harrison, 2006]. It is normally something larger software development teams are doing

[4]Pair programming is an agile software development technique in which two programmers work together at one workstation

as a pre alpha-testing, this will help developers to discover eventual problem-areas early.

1.3.4 Evaluation

The table shown in Figure 1.4 from [Esearch et al., 2004] serve as guidelines for the different evaluations of the application. The evaluations and the results from these are further described in Chapter 9. During the final focus group evaluation we also made use of the evaluation toolbox published by MIRROR[5]. This toolbox is a specification of evaluation methodology and research tooling.

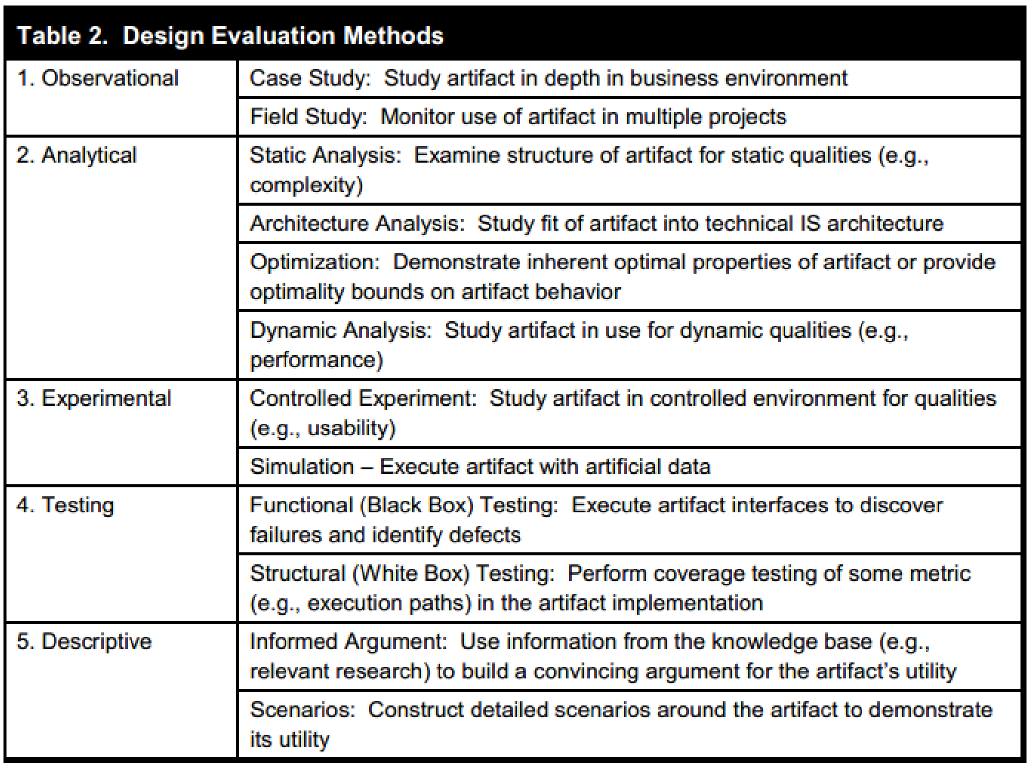


Table 2. Design Evaluation Methods

1. Observational	Case Study: Study artifact in depth in business environment
	Field Study: Monitor use of artifact in multiple projects
2. Analytical	Static Analysis: Examine structure of artifact for static qualities (e.g., complexity)
	Architecture Analysis: Study fit of artifact into technical IS architecture
	Optimization: Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior
	Dynamic Analysis: Study artifact in use for dynamic qualities (e.g., performance)
3. Experimental	Controlled Experiment: Study artifact in controlled environment for qualities (e.g., usability)
	Simulation – Execute artifact with artificial data
4. Testing	Functional (Black Box) Testing: Execute artifact interfaces to discover failures and identify defects
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility
	Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility

Figure 1.4: Design Evaluation Methods

1. **Observational**: Both in the usability study and the focus group evaluation we provided participants with a set of questions where they evaluate the application. In the focus group evaluation we provided

[5]`http://www.mirror-project.eu/showroom-a-publications/downloads/finish/5/67`

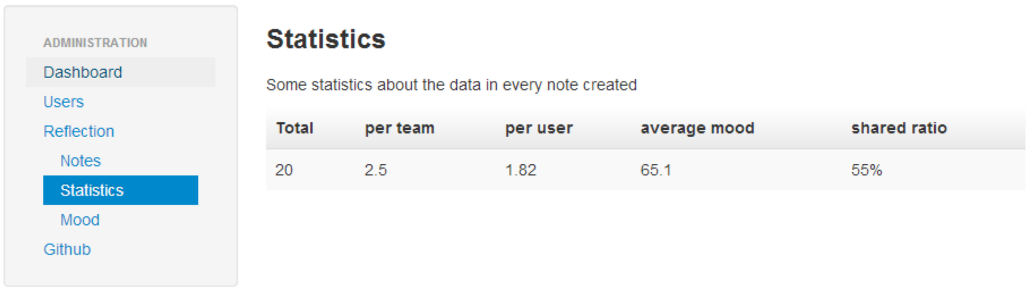
participants with application specific questions and the reflection scale from the MIRROR toolbox [Appendix E]. The reflection scale assesses participants' general tendency to reflect and the importance they place on reflection. Using this scale, allows us to see whether using the tool prime people to reflect more, that is: Does the tool increase the users tendency to reflect on experiences, both individually and as a team.

2. **Analytical**: A separate administration interface were implemented in order to collect data usage statistics from users. These data could provide simple usage-patterns during i.e. a future case-study of a development team using the application. Examples of these data can be seen in Figure: 1.5 and 1.6. Having such data might also provide a basis for an analysis of the application's performance as well as it's usefulness in the domain of reflection in software development teams.

3. **Experimental**: we performed a usability study as a controlled experiment (See Section 9.1).

4. **Testing**: During development we continuously tested the tool to ensure that the user experience is as expected, as a means of quality assurance. White Box testing were used during development of the application with artificial data to ensure functions are working properly and as specified.

5. **Descriptive**: In order to demonstrate the usefulness of our application we constructed two detailed scenarios, demonstrating it's use for tasks in a real work environment (See Chapter 5, Section 5.4). All three evaluations used the context of these scenarios to evaluate the application and provide feedback. In addition to these evaluations we have continuously used the application ourselves in order to identify any problems as we develop the application (See Section 1.3.3).

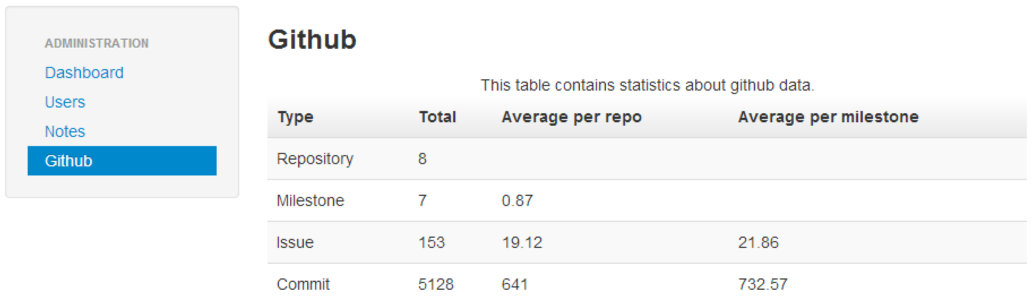


Statistics

Some statistics about the data in every note created

Total	per team	per user	average mood	shared ratio
20	2.5	1.82	65.1	55%

Figure 1.5: Reflection note statistics



Github

This table contains statistics about github data.

Type	Total	Average per repo	Average per milestone
Repository	8		
Milestone	7	0.87	
Issue	153	19.12	21.86
Commit	5128	641	732.57

Figure 1.6: GitHub statistics

1.3.5 Research Contributions

The main goal was to create a proof of concept application that may help users reflect on their past experiences, both individually and collaboratively in reflection sessions, by collecting experiences daily, storing and revisiting them at any time. The implementation of the application and the evaluation of it gave us a foundation to answer our research questions, and also identify areas that can go into future research and development in the domain of technology supported reflection and collaboration. Future work and the identified proposed features and improvements can be seen in Section 10.3 will provide an increased understanding of how technology can support reflection in software development teams, by collecting and revisiting experiences from everyday work.

1.3.6 Research Rigor

The application was evaluated through a usability test, an expert review with an expert in the field of agile software development and a focus group consisting of a software development team. By evaluating the application with experts in the relevant fields provides an indication for the usefulness of the application.

1.3.7 Research communication

The proof of concept application have been released under the GNU Public License v3[6] and the repository are located at GitHub:

- `https://github.com/ekun/PeacefulBanana`

Any limitations identified will be documented along with the research results.

[6]`http://www.gnu.org/licenses/gpl.html`

1.4 Outline

In this section we will describe the organization of the remaining chapters in our thesis.

Chapter 2 gives a general overview of the PeacefulBanana application. The intention is to present features and concepts of the application on an abstract level.

Chapter 3 describes the theoretical background behind over thesis, including reflection, experiences and experience based learning.

Chapter 4 describes our State-of-the-art, the literature review that was performed and related work. The chapter presents some of the work performed in the domains of technology for reflection and experience based learning, with related work acting as a basis for our own implementation of technology.

Chapter 5 presents the problem definition, an introduction to GitHub, user stories and our scenarios.

Chapter 6 describes the requirements chapter, elaborated from problem elaboration and the scenarios.

Chapter 7 describes the design choices behind the creation of the PeacefulBanana tool.

Chapter 8 describes how we implemented the PeacefulBanana tool.

Chapter 9 includes a description of the different evaluations of the application, the usability study, the expert review and the focus group. The chapter also presents a discussion of the results gathered and how they may answer our research questions.

The last chapter of our thesis, is a conclusion with a summary, an evaluation of our own work and ideas towards future work.

Chapter 2

Peaceful Banana - An overview

In this chapter we introduce the PeacefulBanana tool, which is the prototype we developed for our thesis. The purpose of this chapter is to explain the concept of the tool, its features and the functionality it provides. The tool will be presented at a high-level, as the process of how we developed the prototype, design choices and implementation is further described in Chapter 7 & 8.

2.1 What is PeacefulBanana?

Peaceful Banana is a tool aimed towards aiding reflection in software development teams. It integrates with version control systems(VCS)[1], which is commonly used in software development. The PeacefulBanana tool collects project artifacts from such version control systems and scaffolds these in order to trigger and promote reflection in teams. PeacefulBanana integrates with VCS and provides a layer of features specific for aiding reflection in software development projects. I.e. PeacefulBanana will not feature the same functionalities that already exists in these systems, but rather contextualize them and present the data in a manner that can help teams reflect on their experiences. The tool will collect different data relevant for reflection, e.g. prompting the user for their mood each day, which allows for both team-wide

[1]Version control is the management of changes to documents, computer programs, large web sites, and other collections of information

and personal mood-graphs over a period of time.

PeacefulBanana is developed as a web-application and will work in all modern web-browsers running on platforms like PCs/MAC, smartphones and tablets. The tool was developed with project artifacts collected from VCS in mind. The PeacefulBanana tool is developed for use in software development teams that have adopted an agile process model. Most agile process models feature reflection sessions in some form, in which this tool is used in order to promote and enhance the reflection that takes place in these sessions. Although it is aimed at agile teams, any process model featuring some kind of reflection session could benefit from using this tool. Similarly the tool can also be used individually at any time as a more general project overview tool.

The goal of PeacefulBanana is to present software development teams with a tool that can help them revisit and reflect upon their experiences in both an individual and collaborative setting. Through the process of developing the prototype, it has become a tool that also supports collection of project artifacts and the presentation of these to users, in order to connect to experiences and trigger reflection.

2.2 What does it do?

PeacefulBanana is a tool that allows users to reflect upon and share their individual and collaborative experiences from development projects. The tool focuses on collecting project artifacts from VCS like code commits, milestones and issues as well as comments and references. The tool then scaffolds these data and presents them to the users, acting as input for users to revisit experiences and trigger reflection upon these. The reflection that occurs will be captured and stored as reflection notes so the user can review these at a later date. These notes, containing a user's reflection around that day's work experiences, can also be shared with the team and used by the user's team in reflection sessions. The tool is in this way designed to support both individual and collaborative reflection around work experiences. It can be used during the team reflection workshop or just browsed individually when the user wishes to. The tool provides several opportunities to the users:

Provide a scaffolded overview of the project

- Users can choose what team-project to retrieve data from, see the projects milestones, issues and generate data for reflection from these.
- The team can see when an issue has been closed, and what milestones they are connected to.
- The team can see which milestones were closed and when, and easily see if the team met their deadlines.
- The team can see which team members have contributed to which parts of the project.

Individual 5-minute daily reflection

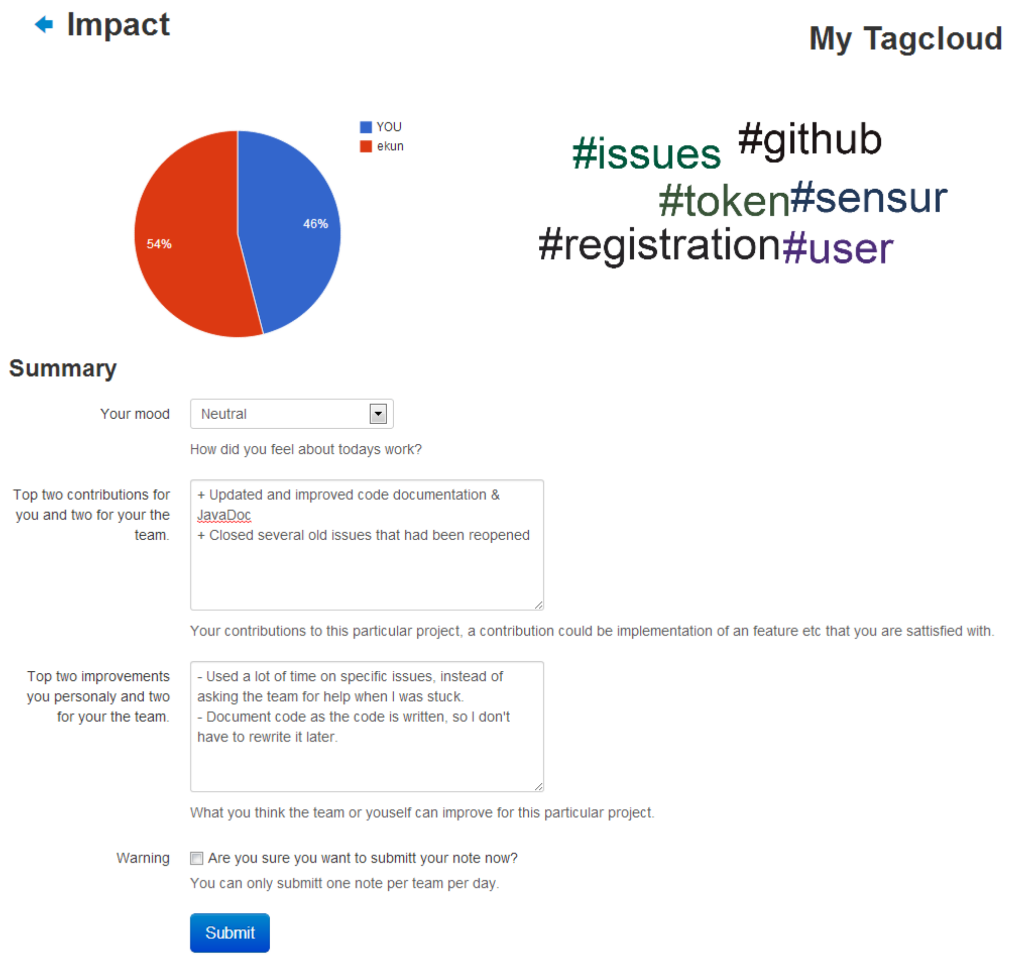
Each day a notification will prompt the user to perform a daily 5-minute reflection. This daily reflection note presents the user with data for the last 24 hours, i.e. project activity and a tag cloud. This data provides the user with additional information and tries to trigger reflection in the users based on that day's experiences. The reflection note collects input from the user:

- The user's mood that particular day, ranging 5 steps from 0 (Very sad) to 100 (Very happy)
- The top 2 contributions done by the user in the project that day.
- The top 2 fields the user can improve on in the project.

Figure 2.1 shows how the daily reflection note looks in the PeacefulBanana application. Each reflection note is stored and can also be shared with the user's team, but sharing notes is not required. The mood data collected each day is used by the application to generate a team-wide mood-average graph. This graph can be used to trigger a discussion and reflection in the team reflection sessions.

Reflection sessions

The team, or the team leader can create a workshop from a selected time-period. The workshop presents some mandatory questions related to team work and reflection. Additionally the team can choose a set of tags to generate questions from. The finished workshop template can be printed and



Your mood: Neutral

How did you feel about todays work?

Top two contributions for you and two for your the team.

+ Updated and improved code documentation & JavaDoc
+ Closed several old issues that had been reopened

Your contributions to this particular project, a contribution could be implementation of an feature etc that you are sattisfied with.

Top two improvements you personaly and two for your the team.

- Used a lot of time on specific issues, instead of asking the team for help when I was stuck.
- Document code as the code is written, so I don't have to rewrite it later.

What you think the team or youself can improve for this particular project.

Warning: ☐ Are you sure you want to submitt your note now?

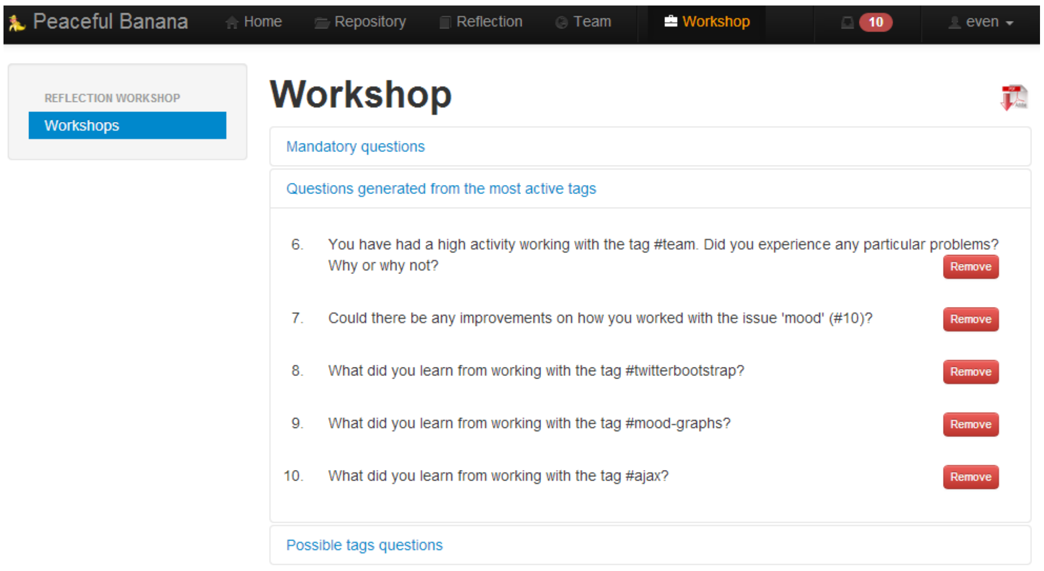
You can only submitt one note per team per day.

Figure 2.1: PeacefulBanana Daily reflection note

handed out to the team at the workshop, providing project statistics, trending issues, tag clouds and questions that act as reflection triggers.
Examples of such questions:

- What were your initial expectations to this iteration? Did these expectations change during the iteration? How? Why?
- What could be done to improve team collaboration?
- Talk about any disappointments or successes of your project. What did you learn from it?
- You have had a high activity working with #framework Did you experience any particular problems with this tag? Why or why not?
- The team didn't meet the deadline for milestone #20 - Midterm Report, did the team experience any particular problems?

Figure 2.2 shows an example of a project workshop and the questions generated for it based on the project tags.



Workshop

Mandatory questions

Questions generated from the most active tags

6. You have had a high activity working with the tag #team. Did you experience any particular problems? Why or why not?
7. Could there be any improvements on how you worked with the issue 'mood' (#10)?
8. What did you learn from working with the tag #twitterbootstrap?
9. What did you learn from working with the tag #mood-graphs?
10. What did you learn from working with the tag #ajax?

Possible tags questions

Figure 2.2: PeacefulBanana Reflection Workshop - Generated Questions based on project tags

Chapter 3

Background

In this chapter we present the theoretical background for this thesis. This includes information on Computer-supported reflection, agile development and the notion of self-organizing teams and a teams importance in agile development.

3.1 Computer-supported reflection

Reflection is critical to workplace learning, enabling employees to make sense of complex and dynamic situations[Schön, 1983]. Boud et al.[Boud et al., 1985] defined learning through reflection as:

> *Those intellectual and affective activities in which individuals engage to explore their experiences in order to lead to new understandings and appreciations.*

The MIRROR project[1], describes reflective learning as:

> *The conscious re-evaluation of experience for the purpose of guiding future behavior, acknowledging the need to attend to feelings, ideas as well as behavior associated with work experience.*[Krogstie et al., 2012]

Work and reflection on this work are heavily connected[Schön, 1983], both driving each other forward. Experiences are created through work, and these experiences can be reflected upon. Reflection can be based on both a memory

[1]`http://www.mirror-project.eu/`

of an experience, and on data from the experience. Revisiting and reflecting on a work experience leads to a better understanding of the experience and allows for learning from it. Reflection can help developers learn from experiences and gain knowledge of how to deal with work-related challenges. This relationship between reflection and learning has been modeled as an experience-based learning cycle[Boud et al., 1985; Korthagen and Vasalos, 2005; Kolb D.A, 1975]. In the reflective process you return to the experience, attend to connected feelings and re-evaluate the experience. These cycles show that the reflective process and the experience connects in order to produce an outcome. Reflection has both individual and social dimensions,[Høyrup, 2004; Woerkom and Croon, 2008]. Social wise reflection is often performed collaboratively by teams performing a joint task as organized practice, and therefore shares some common ground and experience.

Most reflective learning in a work environment, happens analogically without support of technology[Schindler and Eppler, 2003]. Technology has been shown to increase the potency of reflection and reflective learning at work, meaning computer-supported reflection tools can be used to promote reflection and experience based learning [Krogstie and Divitini, 2010; Lin et al., 1999].

3.1.1 The MIRROR model

The MIRROR model of Computer Supported Reflective Learning (CSRL), is a work on how to incorporate reflection in to the daily routine at work. MIRROR presents a reflection-learning cycle based on the CSRL cycle in Figure 3.1, which accounts for the role of technology in reflective learning at the workplace. The MIRROR CSRL model is presented in Figure 3.2, and shows how tools can support the reflection process. One key part of this model is the reflection session. These sessions is where the team gathers and actively reflects on experiences, both informal or formal. The model shows that the reflective process and the experience connects in order to produce an outcome. This result with identified behavioral changes and new perspectives can then be applied to the working environment.

Figure 3.2 depicts the different stages the MIRROR-model introduces. Our thesis builds mainly on the two first phases, the *Plan and do work* and *Initiate reflection* phase. The application developed supports these phases during work, collecting experiences and reflection and making the results available for later use. The application also enables teams and team-members to prepare for the reflection session. Team-leaders can also initiate the reflection

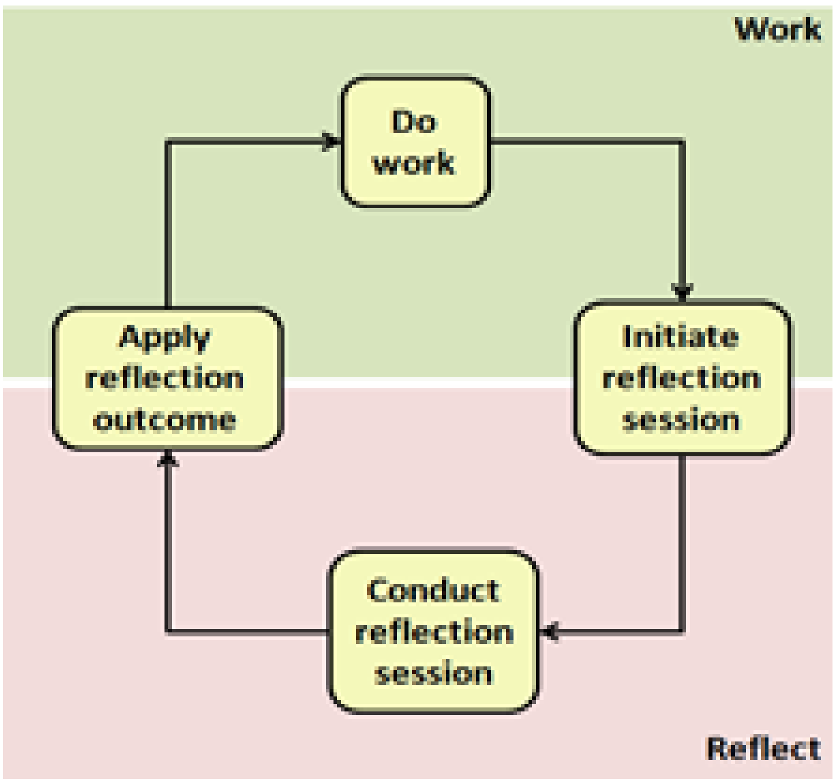


Figure 3.1: CSRL Cycle view [Krogstie and Prilla, 2011]

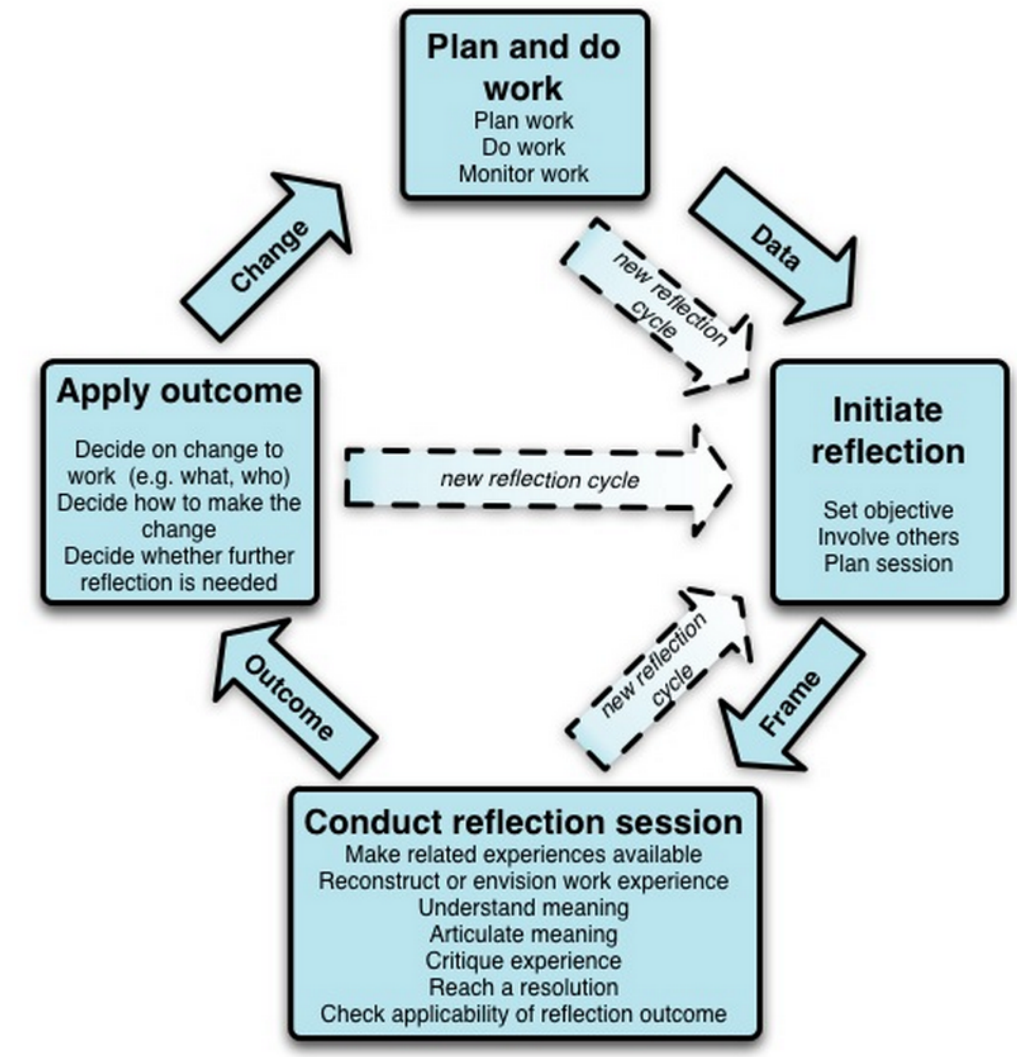


Figure 3.2: MIRROR CSRL model [MIRROR, 2013]

session, by creating a frame or a plan for the team reflection session.

Plan and do work

The *plan and do work* stage is any work related activity. This includes everyday work, planning and monitoring. It also includes simulated work in both real and virtual environments, and the activity can be both individual or collaborative. The data resulting from this phase can be used to reconstruct and make sense of work experiences.

Initiate reflection

The *Initiate reflection* stage is where the reflection cycle starts when reflection has been triggered. The initiation of reflection includes setting the objective for the reflection session and making a plan for it. The result of this phase is a frame to be used during the reflection session. This frame can be seen as the *"setup"* of the reflection session, creating a context for the session. Such a frame allows for an efficient time-use and keeping the discussion to the topics that have been identified in the frame.

Conduct reflection session

The *Conduct reflection session* stage is the reflection session itself, and is based on the frame identified in the *Initiate reflection*stage. The reflection stage will in our context feature agile software development teams in a retrospective reflection session. Activities during this stage use the identified frame to create a discussion around the experience collected by the application and use the reflection outcomes to improve work.

Apply reflection outcome

The *Apply reflection outcome* stage, is where the reflection outcomes are used to create a change in the daily work routine of the teams. This includes what to change and who will be involved in the change. How to make the change, f.ex if it can be immediately applied and if it should be recommended to other teams as well.

We will utilize this model during both the development and evaluation of the proof-of-concept prototype. This prototype will capture experiences and store them for use in reflection sessions. This will make it easy for users to revisit the experience and reflect upon them, be it individually or collaboratively with a team. The challenge will be to apply the correct steps in order to collect the most important information for the different contexts, and discard the less important data. This is vital in order for our tool to collect the behavior, ideas and feelings connected to an experience.

3.2 Agile Software Development

Agile software development consists of several different methods [Abrahamsson et al., 2002], among which Scrum [Schwaber and Beedle, 2002] and Extreme Programming(XP) [Beck and Andres, 2004] are the most used. Out of the two, Scrum has more focus on the project management part of agile development, while XP concerns itself mostly with implementation of software. This thesis is focused on agile teams using Scrum, as Scrum is the method used by our evaluation participants.

Agile projects using Scrum divides projects into iterations called *sprints*, which often are project milestones. Iterations start out with a planning stage and ends with a review during a *retrospective session*. Features that are to be implemented is gathered in a *backlog*, and the implementation order is decided by the product owner. Scrum has set intervals, which normally lasts from 2-4 weeks for each iteration, with each iteration containing roughly the same amount of work [Ken Schwaber, 2011].

Scrum teams are *self-governing* or *self-governed*, which means an "autonomous team" or basically a team that manages itself. Guzzo and Dickson describe self-governing teams as:

> *...teams of employees who typically perform highly related or interdependent jobs, who are identified and identifiable as a social unit in an organization, and who are given significant authority and responsibility for many aspects of their work, such as planning, scheduling, assigning tasks to members, and making decisions with economic consequences*

Such autonomous teams encourage involvement, with team members having an increased commitment and attachment towards the organization and the product they deliver. Also bringing the decision making to the developers, increases the speed of decisions and efficiency, shown by [Tata and Prasad, 2004]. The same authors found that in order to achieve the benefits of an autonomous team, developers need to have an impact on management related decisions, and not just symbolic. Autonomous teams have also been shown to be more productive than more traditional teams [Kirkman and Rosen, 1999]. Scrum teams are given a high grade of responsibility for their own work, including scheduling, planning and decision-making [Schwaber and Beedle, 2002]. Although autonomous teams have a high grade of independence, organizations should provide some sort of control, in order to prevent the teams from sliding out from internal arguments, while still allowing the team to remain agile and unhindered [Takeuchi and Nonaka, 1986].

The Scrum master in a Scrum team, can be seen as the facilitator for the team, but is not an organizer. The team is self-organizing and decides collaboratively on what to do, while the Scrum master can be seen as its protector. The main role for a Scrum master is to enable the team to perform at its highest level, i.e. facilitating meetings, communication with the product owner regarding backlog, and removing any progress obstructions [Schwaber and Beedle, 2002]. The team leader can basically be anyone, but is often filled by a project manager.

Manifesto for Agile Software Development:[Beck, 2013]

> *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.*

Figure 3.3 represents the different iterations a team of developers iterates through when using agile development methodology.

A scrum board as shown in Figure 3.4, is used for each milestone to show which issues have been started on and who is working on which feature. The chart to the right in the figure is a burn-down chart, stating how many hours the team has to work in order to complete on time.

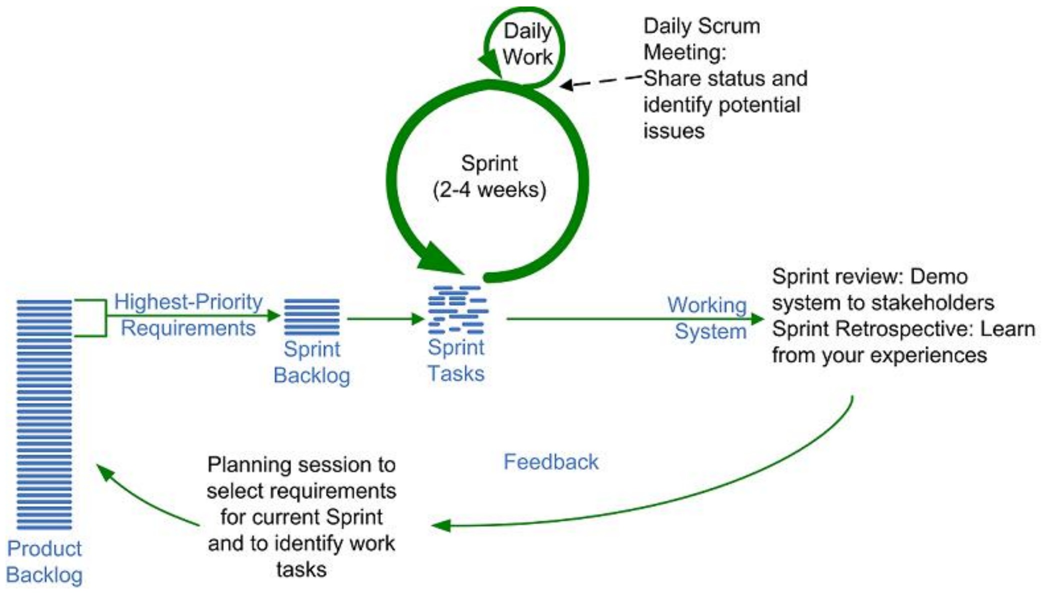

Daily Work
Daily Scrum Meeting: Share status and identify potential issues
Sprint (2-4 weeks)
Highest-Priority Requirements
Sprint Backlog
Sprint Tasks
Working System
Sprint review: Demo system to stakeholders
Sprint Retrospective: Learn from your experiences
Feedback
Planning session to select requirements for current Sprint and to identify work tasks
Product Backlog

Figure 3.3: Agile software development poster [Ambler, 2013]

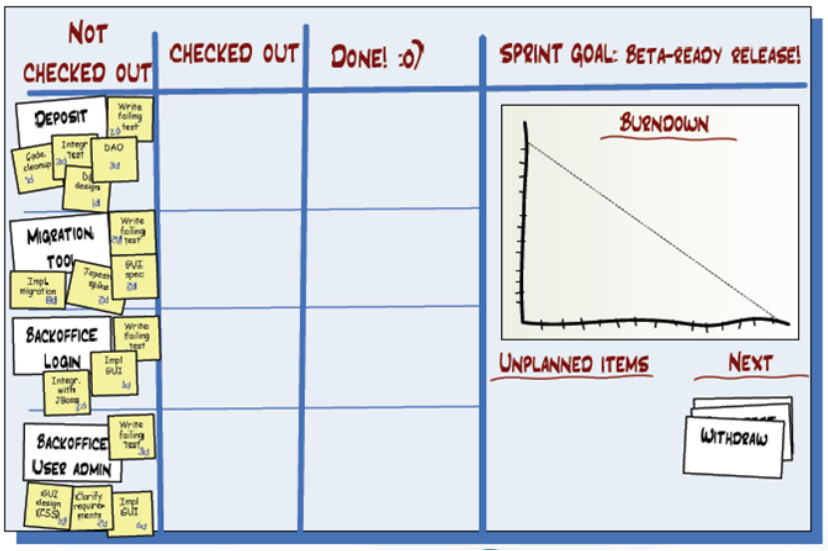


Figure 3.4: Example of scrum board [Kristin Knipfer, 2013]

Chapter 4

State of the art

4.1 Literature Review

In this chapter we will elaborate on the theory on reflection and how technology fits into this domain.

4.1.1 Reflecting on reflection - Important Aspects

D.Talby [Talby et al., 2006] focuses on four aspects proven to be important for reflection. They also proposed a technique for reflection, where the team leader chose the subject beforehand. This has the advantage of not spending time to decide on a subject during the meeting itself, which in turn reduces the time needed. Also there was a reduced risk of selecting 'wrong' subjects, since the moderator(team leader) was a part of the development team and were aware of the day-to-day problems.
Further Talby found that reflection subjects needed to be:

- Relevant to the team in its entirety, and not personal quarrels
- Organizational issues
- Issues there are disagreement on, but not technical problems

Further, open discussions are sufficient to achieve bottom-line results, as opposed to more structured reflection sessions.
A written summary of the reflection session was seen as highly important, even only in an informal way, such as email. This summary was important

in order to exclude any later conflicts on what was agreed upon and in what way.

Reflection levels

There are many ways of using reflection as a concept. Fleck and Fitzpatrick [Fleck and Fitzpatrick, 2010], looks into the purposes of reflections, what is needed for reflection and levels of reflection. Each of these reflection levels capture the activity connected to reflection. The authors state that:

> *...the interest in reflection and technologies to support reflection has expanded beyond these more traditional domains to a range of new areas, with reflection as a topic in its own right.*

Reflection is a time consuming process, which in order to take place needs the appropriate environment. There are five levels identified, where the first level simply describes the situation, and the highest level where the learner sees even farther, taking ethics and morals into account. The findings and techniques of how to support reflection on these different levels [Fleck and Fitzpatrick, 2010], will be used in our development of the PeacefulBanana application.

4.1.2 Tag Clouds

Marti A. Hearst and Daniela Rosner [Hearst and Rosner, 2008] examine the recent information visualization phenomenon known as tag clouds, which are an interesting combination of data visualization, web design element, and social marker. Using qualitative methods, they found evidence that those who use tag clouds do so primarily because they are perceived as having an inherently social or personal component, in that they suggest what a person or a group of people is doing or is interested in, and to some degree how that changes over time. The primary reasons people object to tag clouds are their visual aesthetics, their questionable usability, their popularity among certain design circles, and what is perceived as a bias towards popular ideas and the downgrading of alternative views.

4.1.3 Supporting retrospective reflection in student software engineering teams

Birgit R. Krogstie and Monica Divitini [Krogstie and Divitini, 2009] propose the use of facilitated postmortem workshops in which each team reconstructs its project timeline, in order to help student teams learn from their project process. Individual team member's experience of the project is included by team members drawing individual *'experience curves'* along the timeline. The approach is based on current industry practice and adapted in accordance with theory on reflection and learning.

4.1.4 The functions of multiple representations

Shaaron Ainsworth: Multiple representations and multi-media can support learning in many different ways [Ainsworth, 1999]. In this paper, it is claimed that by identifying the functions that they can serve, many of the conflicting findings arising out of the existing evaluations of multi-representational learning environments can be explained. This will lead to more systematic design principles. To this end, this paper describes a functional taxonomy of MERs. This taxonomy is used to ask how translation across representations should be supported to maximize learning outcomes and what information should be gathered from empirical evaluation in order to determine the effectiveness of multi-representational learning environments.

4.1.5 Agile Project Retrospectives

In this paper we look at the findings of Elizabeth Bjarnason and Björn Regnell about retrospective analysis of agile projects [Bjarnason and Regnell, 2012].Whether it can support identification of issues through team reflection and may enable learning and process improvements. Basing retrospectives primarily on experiences poses a risk of memory bias as people tend to remember events differently which again can lead to incorrect conclusions. This bias is enhanced when looking over a longer period compared to iteration retrospectives. To support teams getting a joint view of projects the article suggests creating a method for visualizing an evidence-based project timeline by illustrating aspects such as business priority, iterations and test activities. The method complements the already existing experience-based approach by providing objective data as a starting point for reflection.

4.2 Related Work

4.2.1 Reflection Approach

CSILE or *Computer Supported Intentional Learning Environments*, is a computer-supported medium created in order to support learning. CSILE is used in [Scardamalia et al., 1989] to show how learning environments can be designed to support reflection. This CSILE system connects multiple computers with a central server, where students can share artifacts like text and pictures in a collaborative setting. Notes are used to share information and experiences, and these notes are later used to compare ideas and perspectives. The system show that students learn both as teams and as individuals, by presenting their personal experiences and reflect upon their own learning by comparing themselves with work done by others in the team.

4.2.2 HackyStat

Hackystat is an open source framework for collecting software metrics in an non-intrusive manner. The Hackystat Framework supports three software development communities:

- Researchers. Hackystat can be used to support empirical software engineering experimentation, metrics validation, and more long range research initiatives such as collective intelligence.
- Practitioners. Hackystat can be used as infrastructure to support professional development, either proprietary or open source, by facilitating the collection and analysis of information useful for quality assurance, project planning, and resource management.
- Educators. Hackystat is actively used in software engineering courses at the undergraduate and graduate levels to introduce students to software measurement and empirically guided software project management.

Hackystat uses sensors in tools e.g. the *Eclipse IDE* to collect data about the developer's activities. Data collected is then used in reports that can be accessed on the hackystat website. The long range goal of Hackystat is to facilitate *"collective intelligence"* in software development, by enabling collection, annotation, and diffusion of information and its subsequent analysis and abstraction into useful insight and knowledge. Hackystat services are designed to co-exist and complement other components in the *"cloud"* of Internet information systems and services available for modern software development.

4.2.3 GitHub tools

In this section we will introduce some tools which are related to GitHub and using project data from GitHub. There are project development tools which integrates with GitHub in order to populate and enhance KanBan or Scrum boards, but these have no features for experience collection and promoting reflection. Most of these tools are aimed at project management and agile development.

GitHub Burn-down App

Made with Node.js, this application is a one of a kind app to create burn-down charts from GitHub issues[1]. An example of a created burn-down chart with this application can be seen in Figure 4.1.

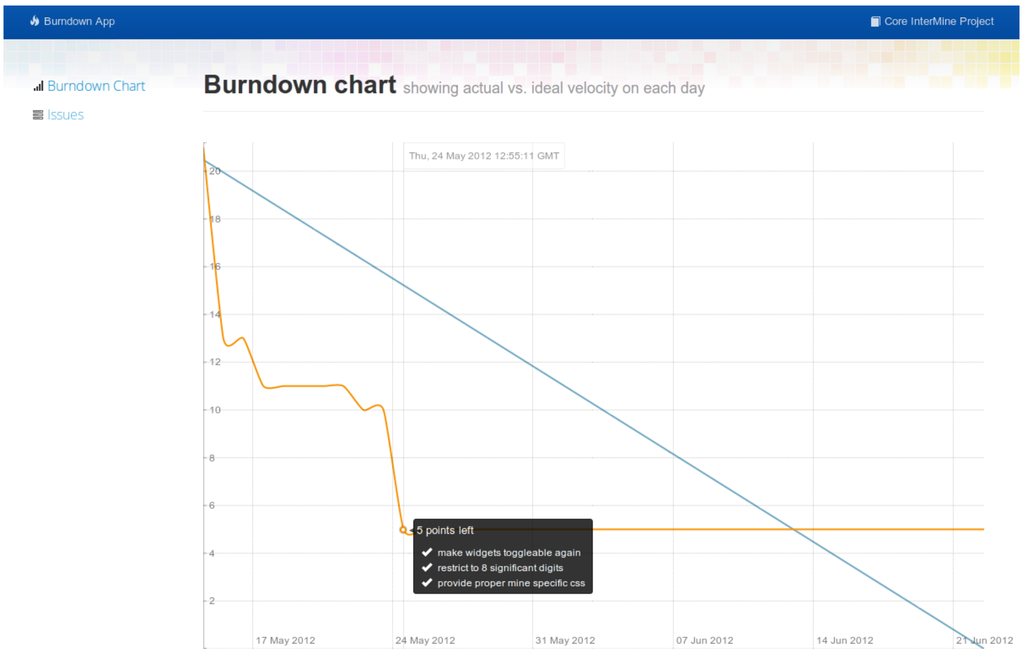

Burndown App
Core InterMine Project
Burndown Chart
Issues
Burndown chart showing actual vs. ideal velocity on each day
Thu, 24 May 2012 12:55:11 GMT
20
18
16
14
12
10
8
6
4
2
5 points left
make widgets toggleable again
restrict to 8 significant digits
provide proper mine specific css
17 May 2012
24 May 2012
31 May 2012
07 Jun 2012
14 Jun 2012
21 Jun 2012

Figure 4.1: GitHub Burn-down application

[1]`https://github.com/radekstepan/github-burndown-chart`

HuBoard

GitHub issues made awesome. [2]
HuBoard is a project management tool, which takes your GitHub issues and milestones and generates a Kanban[3] board built from the GitHub api.
This tool features the idea of scaffolding issues and milestones into something useful for a different context, although it is limited to issues and have no features for capturing experience and promoting reflection.

Agile Bench

Agile Bench integrates with GitHub so programmers do not have to duplicate comments between two systems[4]. Instead they can make a comment via GitHub(the code base) commits and this will push comments into the project management system. AgileBench supports these commit formats:

- `[#story_id]` comment - i.e. "[#5] Added documentation" - will add "Added documentation" to story #5
- `[#story_id #story_id]` comment. i.e. "[#5 #6] Added even more documentation" - will add "Added documentation" to story #5 and #6
- `[Workflow State #story_id #story_id]` comment. i.e. "[In Progress #5 #6] Added even more documentation" - will add "Added even more documentation" to story #5 and #6 and move stories #5 & #6 into the In Progress workflow state.

Agile Bench do not facilitate for experience collection in order to promote reflection.

AgileZen

AgileZen also organizes work in a Kan-ban board, with user-stories[5]. Zen features teams with users and different user-roles. It also allows artifacts like

[2]`http://huboard.com/`

[3]Kanban is a software development method with focus on Just-in-time delivery. Developers pick tasks from a queue. `http://www.kanbanblog.com/explained/index.html`

[4]Agile Bench GitHub integration: `http://support.agilebench.com/entries/21307153-GitHub-Integration`

[5]AgileZen: `http://www.agilezen.com`

screen shots or specification documents to be attached to a story. It integrates with GitHub as a service hook[6]. In order for Zen to include your change sets from GitHub, you need to refer to the story ID in your commits using the format #123 or zen-123. This further builds on the idea of using #hash-tags to *tag* relevant messages and identify them in your commits.

4.3 Discussion

There are several tools that display data collected from VCS like GitHub as described in the previous section, although these tools display data in a different context and with a different goal. Where as we aim to display data for the purposes of connecting these data with experiences to trigger reflection, these tools display data for project management purposes or to gather research data(I.e. *HackyStat*). Common for all the tools introduced in section 4.2.3 are their tight connection with GitHub which allows for efficient data collection.

HackyStat

HackyStat is very general, but aims at collecting usage statistics from software development teams or users. These data are used for reports which are stored for later use to facilitate *Collective intelligence*. We used this for inspiration towards the notion of storing reflection notes for later use, both individually and collaboratively as a team as preparation before the retrospective session.

GitHub Burn-Down Chart

The *GitHub Burn-down* application is a small, yet informative application for development teams using scrum as their agile development process. The application creates a burn-down chart of the current GitHub-project milestone and displays all issues connected to it. The burn-down application has no direct connection to reflection and it's functionality were therefore not implemented, but was used for inspiration when designing the interaction between GitHub and the PeacefulBanana application.

[6] `https://help.github.com/articles/post-receive-hooks`

HuBoard

HuBoard is similar to the GitHub burn-down application in the sense that it connects with GitHub and makes use of project milestones and issues that are present, in order to generate a KanBan board[7]. This tailors the milestones and issues to be used in a better way, more suited for agile development methods. Similar to the GitHub burn-down application discussed above we looked at the integration with GitHub and used that for inspiration on how to best connect and collect data from GitHub through our own application. HuBoard also provided design related inspiration, as the tool is implemented with the *Twitter Bootstrap* framework.

Agile Bench

Agile Bench provided several ideas we wanted to bring to the PeacefulBanana application. Agile Bench uses #-Hash-tags to annotate commits, much in the same way we intended with PeacefulBanana. Also, giving users the possibility to attach feelings and general comments to specific milestones or issues are definitely ideas we wanted to integrate in the application.

AgileZen

AgileZen provided some ideas in the aspect of teams with users and giving the users different roles(i.e. manager or developer). AgileZen also uses #hash-tags to annotate commits, which provided even more justification that hash-tags was a viable way of annotating commits.

[7]Agile process for just-in-time(JIT) production

Chapter 5

Problem Elaboration

In this chapter we will elaborate on our problem by defining our task and presenting our high level requirements. We will also introduce GitHub as the choice of version-control system for our implementation. On the basis of theory, background and related work we will also also present our two main scenarios and the context of these.

5.1 Problem Overview

To answer our research questions we will develop a web-application aimed at software development teams adopting an agile process model. Most agile process models adopt reflection sessions [Kwiecien, 2013], and helping users prepare for these sessions in order to gain the best possible learning outcome is one of the main issues our application try to address. Collecting experience related data throughout a working day, allows users to revisit recent(fresh) experiences and trigger reflection upon these experiences. By capturing these daily reflections and storing them for later use allow users to go back to these experiences and recollect the lessons learned. Sharing these notes with the rest of the team allows the team to collaboratively prepare for the retrospective sessions that take place in agile process models. Reviewing these daily reflection notes individually by the user and/or by the team, further helps identify trending issues spanning the iteration and even the whole project over time. The team can also use the shared notes to compare experiences and trigger discussion and reflection upon these.

Most of these teams use some sort of a version control system with project

artifact's, like programming code, text documents etc. We have chosen to focus on GitHub(See section 5.2), but the application could be adapted to work with any version tracker that allows for data collection.
Figure 5.1 shows the overall process of our application, how user's interact with GitHub and the PeacefulBanana application.

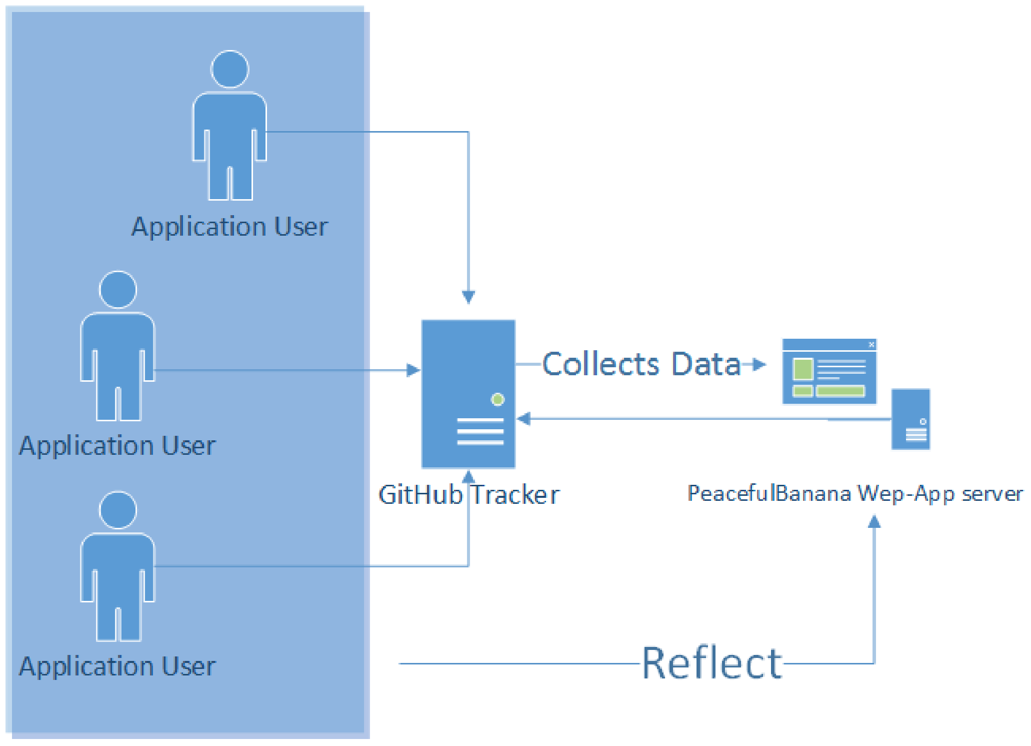


Figure 5.1: Overall application process

The application will collect relevant data from the project repository on GitHub without any action needed from the user to do so. The application will then *scaffold*, that is present the collected data in a structured frame in order to allow users to revisit and reflect upon agile project experiences, and also share these reflection notes.

The experiences will be presented in several different ways like activity-graphs, mood-graphs and tag clouds. These will act as reflection triggers and help the users to reflect on their work. These experiences will be collected and made available for the users and/or the team later. The application will be designed with reflection in mind, and scenarios was created to demonstrate the most important functionality regarding enhancing reflection in software development teams. Although we evaluate the application in the context of demonstrative scenarios, we see the possibility of the application being used outside of the context provided by scenarios. Users can at any time look into a repository, a milestone or a single issue and the application will provide data that is useful for recollecting experiences and providing a basis for reflection upon these.

We have worked out two main scenarios, which can be seen in Section 5.4. Evaluation of the application will be conducted in the context of these in order to gain feedback on how the application may answer our research questions. Evaluation will be a usability test, an expert review and a focus group.

5.2 Github

This section will introduce GitHub [GitHub, 2013a], the features GitHub provides and the rationale behind choosing GitHub as the VCS for our application implementation. GitHub is a web-based hosting service for software development projects that use the Git revision control system [Git, 2013]. GitHub offers both paid plans for private repositories, and free accounts for open source projects.

GitHub provides users with integrated issue tracking, code review, project wiki, useful statistics and more. Motivational points for choosing GitHub as the revision control system to integrate with are several. Some technical aspects are that GitHub provides developers with a simple to use API [GitHub-API-v3, 2013], with libraries for integration with most of the commonly used programming languages, like Java [JGit, 2013]. In addition to the technical aspects, GitHub is the most used revision control system ,as of January 2013 GitHub announced it had passed the 3 million users mark and now hosting more than 5 million repositories and is a application many in our evaluation group already use in their development projects [GitHub, 2013b].

5.2.1 Authentication

In order to retrieve data from repositories on GitHub, GitHub users need to allow the specific application access to their repository. This is done by authentication, which is a feature offered from GitHub to external applications through their GitHub APIv3.

5.2.2 Repositories

GitHub is a repository hosting service for software development projects. A repository contains all the project files, may it be code, images, and other documentation. This means that all content in a project is connected to this repository. In addition to documentation and file content, GitHub provides integrated issue tracking, which is connected to this repository and could also be connected to one or several milestones.

5.2.3 Commits

Say some of your project files have been changed, for example some code snippet in a Java file. The way to save these changes to your local branch is by commits. A commit can consist of code line additions, deletions, files added, changed or removed. When you are ready to save the changes, you commit them in a command line together with a commit message which is a short text describing the changes you have done.
When a user is ready to push one or more local commits to the project repository, it can be done via the command git push in the command line. It is now the HEAD[1] revision and the new code and commits can be seen on the GitHub page.

5.2.4 Milestones and Issues

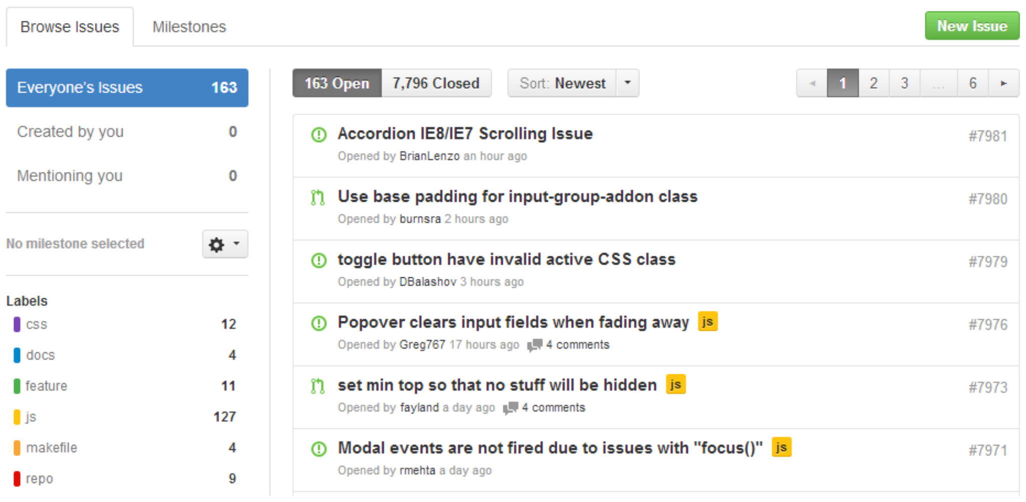
GitHub Issues can be assigned to a user to make it easy to know who's working on what, or which issues you need to tackle next.

Every GitHub repository has an issue tracker, that allows users to track bugs and focus on features. An example of such an issue tracker can be seen in Figure 5.2, which depicts the current issue tracker of Twitter Bootstrap's public repository[2]. Milestones and issues help manage large projects, where issues especially makes for a great TODO list, similar to a product backlog. Only collaborators[3] can create and view issues on private repositories. On public repositories anyone can create and view issues.

[1]Git uses the HEAD variable, which by default, is a reference to the current (most recent) commit.

[2]`https://github.com/twitter/bootstrap/issues`

[3]A collaborator on GitHub is a user who have contribution-rights for a specific repository.



Labels

css	12
docs	4
feature	11
js	127
makefile	4
repo	9

Accordion IE8/IE7 Scrolling Issue — #7981
Opened by BrianLenzo an hour ago

Use base padding for input-group-addon class — #7980
Opened by burnsra 2 hours ago

toggle button have invalid active CSS class — #7979
Opened by DBalashov 3 hours ago

Popover clears input fields when fading away js — #7976
Opened by Greg767 17 hours ago 4 comments

set min top so that no stuff will be hidden js — #7973
Opened by fayland a day ago 4 comments

Modal events are not fired due to issues with "focus()" js — #7971
Opened by rmehta a day ago

Figure 5.2: Example of GitHub's issue tracker overview [GitHub.com-TwitterBootstrap, 2013].

- GitHub issues can be assigned to a user, which makes it easy to know who's working on what, or which issues should be handled next. Milestones are a good way of helping team members to work towards a goal. A team can set a due date, name a milestone and then start assigning issues to that milestone. An example of a Milestone could be a due date of a project demo or delivery. Any number of issues can then be assigned to this milestone, and thus be connected to it.

- Issues know all about commits. GitHub enables referencing and closing issues with Commit Messages. By using a few simple keywords you can close an issue right from a commit message, or just leave a note on the issue.The syntax to do this is as follows: To close issue #35 , a commit message containing 'closes #35' , will close issue number 35 when pushed to GitHub. Other keywords are: close, closes, closed, fixes, fixed.

  To leave a note on issues can be done by simply mentioning the issue number without any keywords in a commit message. F.ex "This commit references #35". Anyone with write access to a repository may close an issue or leave a note.

5.3 Using technology to promote learning from reflection

The world wide web and modern technologies provides easy access to enormous amounts of information. This means that in order to learn, learners must be able to make sense of the information collected. In order to achieve this and make conscious decisions of how to use information, learners need to reflect on the information they collect. Reflection upon the process of solving problems is necessary to achieve a good result and to improve the ability to learn from experiences. When supporting learning with technology, this technology should promote these aspects within learning[Lin et al., 1999].

By implementing this proof of concept application, we wish to provide technology that enables efficient information retrieval, and to provide scaffolds or structured frames that support reflective thinking and problem solving. In our development of PeacefulBanana this means utilizing both individual and collaborative learning experiences.

5.3.1 Agile Retrospective

The agile retrospective is held at the end of an iteration in order to learn from the iteration and not repeat mistakes. A model of how a retrospective can be done is shown in [Derby and Larsen, 2006]. The purpose of the retrospective is to learn what works and what does not work, and make the adjustments necessary for the next iteration. This way the team makes sure that every iteration introduces some improvements in the team's process. There are two fundamental questions the retrospective is intended to answer:

- What went well during the last iteration that we continue doing?
- What could we do differently in order to improve?

These questions are something we incorporated into both the individual and collaboration parts of the PeacefulBanana application, since they are a vital part of the learning outcomes in agile retrospectives. The application primarily aims at providing a structured frame for the retrospective, although all parts of the retrospective needs to be considered during development. Figure 5.3 shows where the retrospective fits in the overall agile iteration cycle, and the table in Figure 5.4 shows the retrospective process in more details [Derby and Larsen, 2006].

Knowing the process of the agile retrospective was important during development. The PeacefulBanana application provides teams with a frame for the retrospective, containing the most relevant issues the team should discuss. This fits with the *Set the stage* phase in Figure 5.4. This frame also brings the team a shared and collaborative setting, since it contains the most commonly worked on issues for the whole team. This covers the *Gather Data* and *Generate Insights* phase. Additionally user's can use the application to prepare individually for the retrospective. Having these individual points of view may give the team greater insight to issues that may have been lost in the collaborative setting. I.e.if a user's work differs a lot from the rest of the team, this may indicate missing competence-overlap and needs to be discussed.

The application provides the team with a printable template or frame they can follow during the retrospective session, which they can use to write down their thoughts on each of the issues the team discuss. Since a vital part of the retrospective session is to write down the learning outcomes, the printable frame allows the team to collect this, covering the last two phases in a retrospective. Since the retrospective session is conducted analogically, these answers are written down on paper, and is not collected by the system.

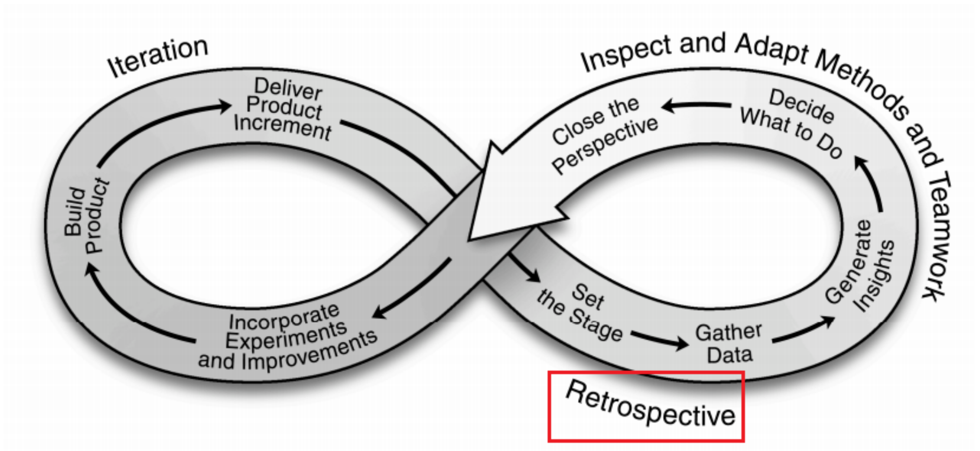
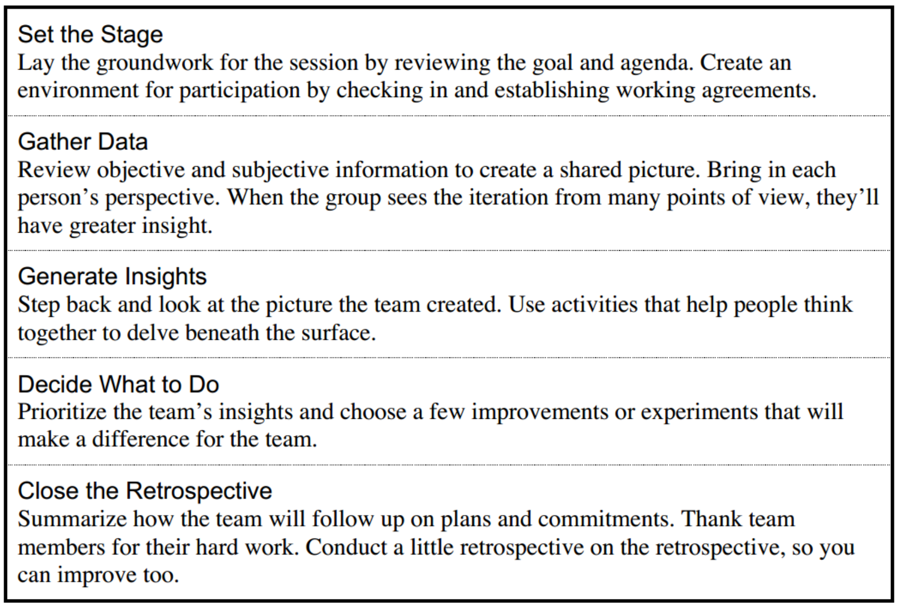


Figure 5.3: Retrospectives in the agile iteration cycle [Derby and Larsen, 2006]


Set the Stage
Lay the groundwork for the session by reviewing the goal and agenda. Create an environment for participation by checking in and establishing working agreements.

Gather Data
Review objective and subjective information to create a shared picture. Bring in each person's perspective. When the group sees the iteration from many points of view, they'll have greater insight.

Generate Insights
Step back and look at the picture the team created. Use activities that help people think together to delve beneath the surface.

Decide What to Do
Prioritize the team's insights and choose a few improvements or experiments that will make a difference for the team.

Close the Retrospective
Summarize how the team will follow up on plans and commitments. Thank team members for their hard work. Conduct a little retrospective on the retrospective, so you can improve too.

Figure 5.4: Agile retrospectives - from beginning to end [Derby and Larsen, 2006]

5.4 Scenarios

Based on related work, we have developed two scenarios to show what we want our application to support in terms of collaborative learning from reflection. These two scenarios will provide a basis for the requirement elaboration and design choices in development of the tool. The application is developed to be used by agile software development teams in a real working environment. Further we developed the application with teams using GitHub as a version-control system in mind. During the project work the team will use GitHub to collect information. The PeacefulBanana application can then be used to gather this information and present it in a scaffolded, or structured way, mainly to enable user to trigger reflection individually and to used in retrospective sessions. Users can though, at any time go into the application and gather relevant data if they want to. This data will help the team see trending issues and problems they have come upon in the process. Each member of the group will register as a user on the PeacefulBanana application and the users will then be connected together as a team, if they are part of the same team on GitHub.

The two scenarios described in the next sections provides a demonstration of how we envision the usage of the PeacefulBanana application in the context of different settings. First as a individual application on a daily basis, then as part of a team collaborative reflection session. These two scenarios were developed early in the development process. The first scenario features using the application at the end of each working day as inspired by the 5-minute daily reflection(See Section 2.2). The second scenario is set in the context of that in each iteration there is a retrospective reflection session at the end.

5.4.1 Scenario 1 - Individual use on a daily basis

In this scenario, our team users will be using the application on a daily basis at the end of each working day. When users enter the web-application, they will get a notification with a prompt to do the daily status update.

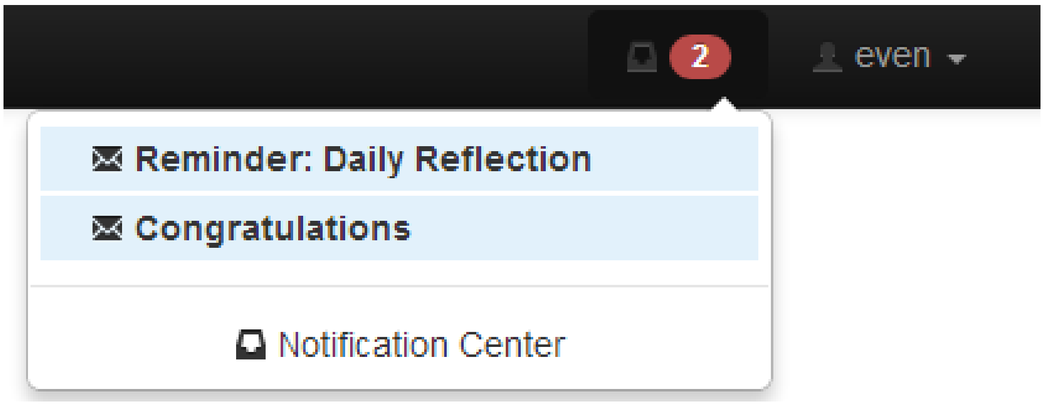


Figure 5.5: The user has clicked the notification icon and can see his new notifications

When clicking the notification, the user is presented with a summary of their individual activity in the last 24 hours. This summary compares the commit activity of the user with the team's, and also presents a tag cloud representing the trending issues of the last day. The user is then prompted to input todays mood, their top two contributions *"What did I do good?"* , and their top 2 points to improve on *"What could I do better?"*. Finally the user can submit the form and choose to share these experiences with the team for collaborative use. The daily reflection notes are saved and can be reviewed at any time, and also shared any time by the user. The idea is that the application will be used to revisit todays experiences with collected information presented in a way that triggers reflection upon these experiences. Information is collected and inspire new experiences. After working on a project that day, users ”step back” to reflect on what happened during the day, and on the experiences they encountered. The application provides scaffolded data to further trigger this reflection process. This process consist of returning to the experiences of that day, re-visit the experiences and attend to the feelings, like inputing todays mood. This reflection will help users to derive their top two improvements and contributions of the day[Krogstie and Prilla, 2011]. The application thus captures the experiences and reflections made by the user and allows for re-visiting of these at a later date by storing them in the system.

5.4.2 Scenario 2 - Team use after each iteration

In this scenario, users will be using the application as part of the preparation for the agile retrospective sessions. As a requirement users need to use the application as described in Scenario 1 (Section 5.4.1). This means going through the daily reflection note process each day they have contributed to a project. Based on the notion that iterations in agile teams often last for 30 days or less [Ken Schwaber, 2011], the scenario was created with the aspect of iterations lasting at least two weeks each. This way enough data can be gathered through the reflection notes, and experiences are still fresh.
Users will be able to individually prepare for these retrospective sessions, while the team can use the application collaboratively as preparation for the session as well. Users will in this scenario use the application to indicate how the project has been progressing over the last iteration, tightly coupled with one or several milestones. They will be able to generate tag clouds based on the trending issues in the relevant milestones, activity graphs and mood trajectories. Examples of such a tag-cloud and mood-graph can be seen in Figure 5.7 and Figure 5.6. Also if the users have chosen to share any of the individual reflection notes from scenario 1, these will be visible and can be used by the team as a whole to draw conclusions from the previous iteration, create a discussion and make comparisons. The application will enable the team to see the whole iteration more clearly, but also enable teams to dive into certain issues or milestones that showed to be of particular interest, and thus create a discussion around the experiences made by the team members.

As a final part of the scenario, teams will be able to create a *workshop* based on any previous iteration period. When the workshop has been created, the team manager or the team as a whole get a set of system generated questions that relate to the workshop iteration. These data are gathered from the team's reflection notes and general GitHub data, and try to present what issues the team have been working the most on for the iteration. The team will, after selecting the questions or topics they want to discuss in the retrospective session, be able to print these as a guide the team can follow during the session. Their reflection outcomes or lessons learned can be noted on the paper by each team member, together with thoughts or comments.

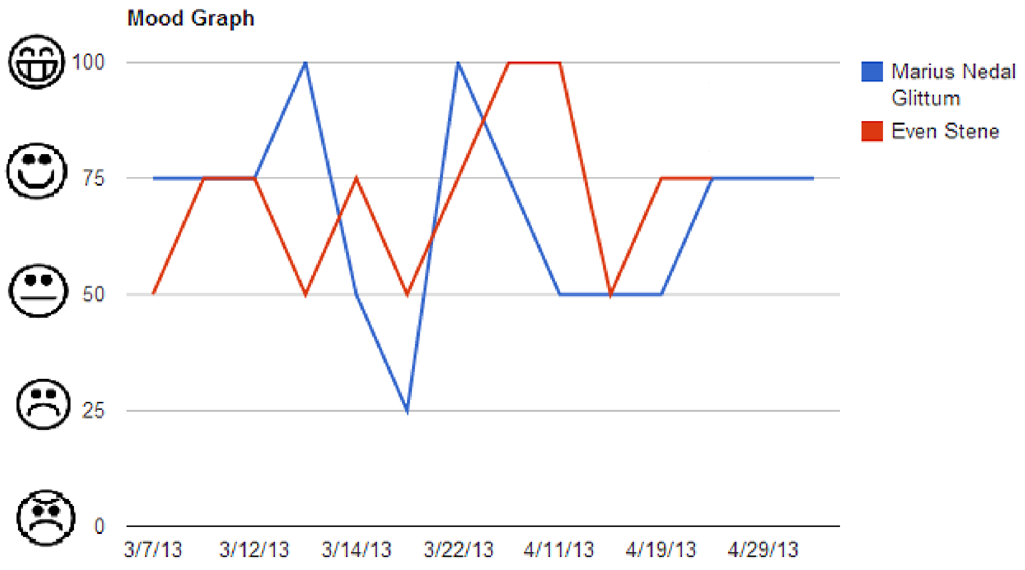


Figure 5.6: Example of a team mood-graph over a period of time.

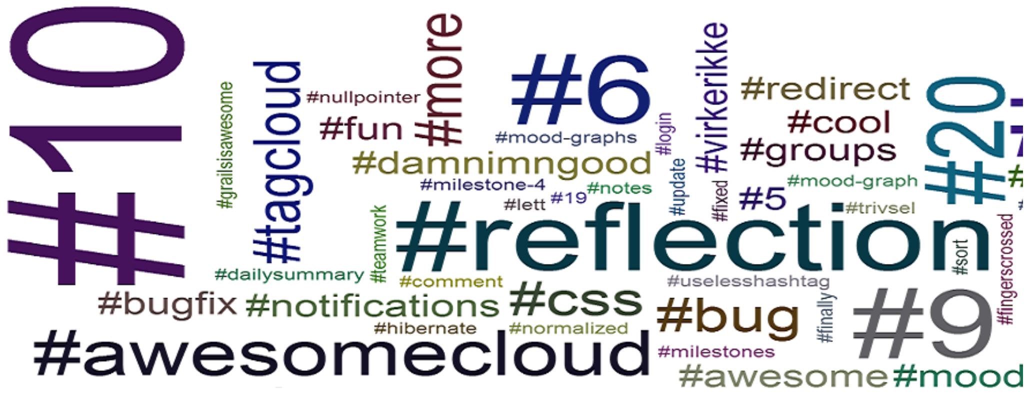


Figure 5.7: Example of a team tag-cloud.

Chapter 6

Requirements

This chapter describes both functional and non-functional requirements the application should meet, in order to better support elaboration and implementation of the final application. Because of the time limitation, we did not expect to be able to implement every requirement, although all requirements were still a viable implementation in terms of the goals for this thesis. Since some of the requirements build on and is dependent on each other, the requirements described were used as a guide when implementing and not as a strict top-down list.

6.1 Functional Requirement

The requirements detailed in this chapter are prioritized according to how important they are for the implementation, with a three step scale.

H High-priority requirements must be met by the application.

M Medium-priority requirements should be met by the application, but omission will severely limit the application.

L Low-priority requirements is not vital to the application, but should be met if all high and medium requirements are met.

For this thesis, a three step scale provides us with adequate data for decisions without sacrificing too much fidelity. In addition to prioritizing the requirements based on importance, the requirements will also be prioritized according to how hard and time consuming they are to implement. These priorities will use the H-M-L scale as described below.

H High is something that will take more than 10 hours to implement.

M Medium is something that will take between 5 and 10 hours to implement.

L Low is something that will take less than 5 hours to implement.

Annotating requirements with both importance and difficulty priorities provides a a way to perform a *cost/benefit* analysis for each of the requirements. A cost/benefit analysis is a way of weighing the total value of benefits against the costs of implementing them[Cellini and Kee, 2010].

$$NetBenefits = TotalBenefits - TotalCost \tag{6.1}$$

A requirement that is hard to implement fully, but is of little importance to the project as a whole could be dropped in benefit for a more important one (or a less difficult one), if time or costs involved are deemed to be to large.

6.1.1 General Requirements

The requirements in this section are the most general requirements for the PeacefulBanana application. These are related to the most common functionality users encounter during use, i.e. getting started with the application and logging in. All of the general requirements are set as *H - Essential*, meaning they have a high importance, since all other requirements depend on these.

FR1 Users must be able to register a new account.
An account is necessary in order to use the application.

Impact	**Values**
Importance priority	**H** – Essential
Difficulty	**M** – Implementation-time 5-10 hours

FR2 Users must be able to log in
An active session is necessary in order to present user-specific data

Impact	Values
Importance priority	**H** – Essential
Difficulty	**M** – Implementation-time 5-10 hours

FR3 Change password
As a [User] I want to [be able to change my password]

Impact	Values
Importance priority	**H** – Security wise, the ability to change password is very important
Difficulty	**L** – 5 hours

FR4 Reset password
As a [User] I want to [be able to reset my password] if I forget it.

Impact	Values
Importance priority	**H** – If a user looses his password, a safe recovery procedure is important.
Difficulty	**M** – Reset-password functionality: 10 hours

6.1.2 GitHub Requirements

The requirements in this section are related to what the application needs in order to collect satisfactory data from GitHub. This includes authorizing the application for use on a user's GitHub account and synchronization.

GR1 GitHub account authorization
The application should be able to authorize the application with users GitHub account

Impact	Values
Importance priority	**H** – Essential as users need to allow the application to access data from their GitHub repositories
Difficulty	**H** – Could be very time consuming, 10-20 hours

GR2 GitHub Synchronization
The application should be able to synchronize with GitHub whenever there is a new event or commit in one of the user's repositories.

Impact	**Values**
Importance priority	**H** – Essential. Synchronization is important in order to present the newest data to the users
Difficulty	**H** – Major functionality. 20-30 hours

GR3 GitHub synchronization status
As a [User] I want to [be able to see my GitHub synchronization status] so I can see if I need to authenticate or if I haven't set a team.

Impact	**Values**
Importance priority	**M** – Moderately important, gives users feedback over their GitHub synchronization status and lets them know if any action is required.
Difficulty	**L** – 5 hours

GR4 Force GitHub synchronization.
As a [User] I want to [be able to force a GitHub synchronization] so that I can ensure all the newest data are synchronized from GitHub.

Impact	**Values**
Importance priority	**M** – Changes on GitHub is automatically detected and updated by the application, but if this synchronization does not occur, allowing users to force it enables them to be sure that the newest data is included.
Difficulty	**L** – 5 hours

Team Requirements

These requirements are concerned around connecting user and team specific features in the application with GitHub. This enables users to create a team consisting of users from their GitHub project, changing their roles on the PeacefulBanana application and joining other teams the user is a part of. The notion of a team is necessary in order to make use of the sharing of reflection notes, as mentioned in both scenarios in Section 5.4.

TR1 Create a team
As a [User] I want to [create a team] so my fellow collaborators can join the team.

Impact	**Values**
Importance priority	**H** – Essential, a team need to be created for users to join it.
Difficulty	**M** – Moderately time consuming. 10 hours

TR2 Join a team
As a [User] I want to [join a team from a list of available teams] so I can share and see shared reflection notes and other team relevant data.

Impact	**Values**
Importance priority	**H** – Essential, a team is required towards the collaborative and cooperative functions of the application, like reflection note sharing.
Difficulty	**L** – 5-10 hours

TR3 Inspect my team
As a [User] I want to [inspect a team] so I see active members and their role in the team.

Impact	**Values**
Importance priority	**H** – Essential, users need to be able to see who is in their team in order to identify if someone is missing or an uninvited person is on their team.
Difficulty	**L** – Moderately time consuming. 5 hours

TR4 Change team role
As a [Team Manager] I want to [be able to change the role of team members] so that I can ensure that the members have the correct user-role in the application.

Impact	**Values**
Importance priority	**H** – Essential, as only a team manager can access certain restricted functionality in the application.
Difficulty	**M** – Moderately time consuming. 10 hours

Milestone Requirements

These requirements are related to both individual and collaborative use, as part of a preparation for the retrospective sessions mentioned in scenario 2, Section 5.4.2.

MR1 Active Milestones
As a [User] I want to [see what milestones are being worked on] so that I can see if the issues related to these milestones.

Impact	**Values**
Importance priority	**M** – Filtering milestones by their status allow users to see what milestones are unresolved and active, and find issues connected to these.
Difficulty	**M** – Moderately time consuming. 5-10 hours

MR2 My closed issues
As a [User] I want to [see what issues have been closed] so that I inspect the issue and recollect how we solved the specific issue.

Impact	**Values**
Importance priority	**M** – Allowing revisiting of resolved issues gives users a chance to revisit and learn from previous mistakes or right-doings.
Difficulty	**M** – Moderately time consuming. 5-10 hours

6.1.3 Architectural Requirements

The application is designed with a client-server architecture in mind, so the requirements here are divided to the specific parts of the architecture. In this case the client requirements concerns the application's graphical user interface and the server requirements concerns the application's back end functionality.

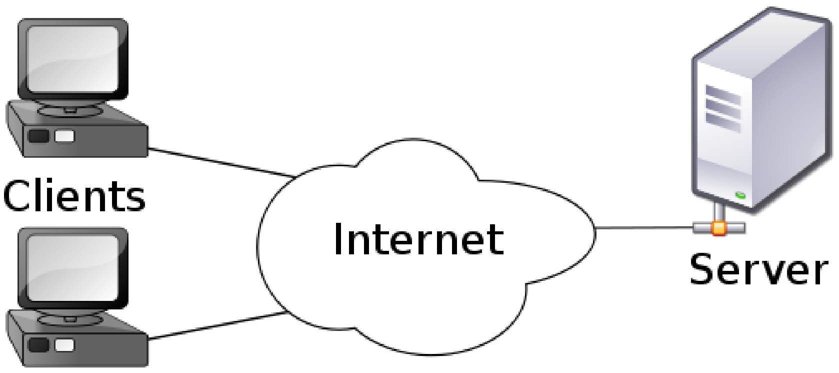


Figure 6.1: Client-server model [Robin, 2011]

Client Requirements

These requirements are for the application graphical user interface and the functionality they need to have in order for them to perform the scenarios defined in section 5.4. Especially the core functionality featured in the *Daily Reflection Note* and the sharing and inspection of these.

CR1 Reflection Notes
As a [User] I want to [save my daily reflection notes] so I can review them at a later time.

Impact	**Values**
Importance priority	**H** – Essential, in order to support reflection, users need to be able to revisit experiences
Difficulty	**M** – Moderately time consuming. 5-10 hours

CR2 Connect feelings & mood with reflection notes
As a [User] I want to [connect my mood & feelings to a reflection note] so I can revisit the feelings in the retrospective sessions.

Impact	Values
Importance priority	**H** – Essential, connecting a users mood to a specific experience help users revisit the experience
Difficulty	**M** – Moderately time consuming. 5-10 hours

CR3 Tag-cloud
As a [User] I want to [see a tag cloud of my most active tags] so I can remember the situations I worked most on and reflect upon them.

Impact	Values
Importance priority	**H** – Essential, arranging the most active tags in a tag cloud will help users easily identify and separate the important situations from the less relevant ones
Difficulty	**H** – Highly time consuming. 10-15 hours

CR4 Tag cloud: Personal vs Team
As a [User] I want to [compare my personal tag cloud with my team's tag cloud] so I can compare my issues with the teams trending issues

Impact	Values
Importance priority	**H** – Essential, comparing personal tagcloud with the team's help identify if the team's trending issues differ from yours
Difficulty	**H** – Highly time consuming. 10-15 hours

CR5 Commit Impact
As a [User] I want to [see my commit impact] so I can compare my activity with the team's activity.

Impact	Values
Importance priority	**M** – Essential, gives users a general idea of who is contributing to the project
Difficulty	**L** – 5-10 hours

CR6 Issues
As a [User] I want to [see my team's issues for the different milestones] so I can see the status of each milestones' issues.

Impact	Values
Importance priority	**H** – Essential, gives users a general idea of who is contributing to the project
Difficulty	**M** – Moderately time consuming. 10 hours

CR7 Inspect Specific Issue
As a [User] I want to [inspect individual issues in a specific milestone] so I can see the status of the issue and what events that are related to this issue.

Impact	Values
Importance priority	**H** – Essential, gives users a general idea of who is contributing to the project
Difficulty	**M** – Moderately time consuming. 10 hours

CR8 General Issues
As a [User] I want to [see my projects general issues] so I can see what is being worked on outside of particular milestones.

Impact	Values
Importance priority	**M** – Essential, gives users a general idea of who is contributing to the project
Difficulty	**M** – Moderately time consuming. 10-15 hours

CR9 Share Reflection Notes
As a [User] I want to [share my reflection notes] so I can share my experiences with the team.

Impact	Values
Importance priority	**H** – Essential, users need to be able to share their notes with the team.
Difficulty	**M** – Moderately time consuming. 10 hours

CR10 Sort Reflection Notes
As a [User] I want to [sort my reflection notes by 'Shared' status] so I can filter out my personal reflection notes from the team's.

Impact	Values
Importance priority	**M** – Moderately important, filtering enhances the user experience, but is not essential for the functionality of the application.
Difficulty	**L** – 5 hours

CR11 Reflection Note reminder
As a [User] I want to [be reminded to do my daily reflection] so I always reflect and collect my experiences of that particular day.

Impact	Values
Importance priority	**M** – Essential, in order to support reflection, users need to be able to revisit experiences
Difficulty	**L** – 5 hours

Server Requirements

These requirements are designed to ensure that the application works as intended in the scenarios in section 5.4 and that the data is handled in a satisfactory manner. This includes a safely-encrypted persistent storage solution, MySQL and the required domain classes.

SR1 Persistent Storage
Create MySQL database support for creating and performing actions on persistent databases by the application

Impact	Values
Importance priority	**H** – Essential, persistent storage is vital for the application to collect and store data
Difficulty	**M** – Moderately time consuming. 5-10 hours

SR2 Databases
Create databases for different functionality: users, reflection notes, commits, milestones, repositories, etc

Impact	Values
Importance priority	**H** – Essential for functionality that requires synchronization with database
Difficulty	**M** – Moderately time-consuming. 8-10 hours

SR3 Domain Classes
Create domain classes for the relation database

Impact	Values
Importance priority	**H** – Essential for all entities and their relations in the database
Difficulty	**H** – Could be very time consuming, 10+ hours

6.2 Non Functional Requirements

This section presents the non-functional requirements for the PeacefulBanana application.

6.2.1 Usability

According to Bass, Clements and Kazman, *usability* is concerned about making the the tasks in the system as easy to accomplish as possible [Bass et al., 2003, p. 90]. Below we have identified use cases in order to assure satisfactory usability of the application.

U1 Support multiple devices
The users should be able to access the application on any device and screen size.

	Values
Source	Developer
Stimulus	Make design responsive
Artifact	System
Environment	Run-time
Response	Scale after resolution
Response measure	The application shall fit the device's screen size

U2 Intuitive design

The user interface must be easy to understand and not to create any confusion.

	Values
Source	Developer
Stimulus	Make user interface easy to use
Artifact	System
Environment	User interface
Response	Place buttons in a 'natural' matter.
Response measure	Users will find the link they are looking for more than 95% of the time.

U3 The application shall show hints where ever appropriate.

The user gets well-founded recommendations, tips or warnings during use, in order to increase confidence.

	Values
Source	End User
Stimulus	User is uncertain on how the application is used, or what their next move can be
Artifact	System
Environment	Run-time
Response	A message box with tips
Response measure	User can use the application without errors regarding the application's usability in *1 hour*

6.2.2 Availability

Availability is concerned with system failure and its associated consequences. A system failure occurs when the system no longer delivers a service consistent with its specification[Bass et al., 2003]. Below we have identified the following use cases for assuring the availability of the application.

A1 Uptime

The application shall be available more than 90% of the time.

	Values
Source	End User
Stimulus	Accessing website
Artifact	System
Environment	Website
Response	Show the user the website it desires.
Response measure	Web server will be working correctly more than 90% of the time.

A2 Persistent storage
When the system is rebooted it needs to restore to the same state as before.

	Values
Source	End User
Stimulus	Accessing website
Artifact	System
Environment	Website
Response	Show the user the website it desires.
Response measure	After shutdown the application shall recover to the same state as before every time.

6.2.3 Security

According to Bass, Clements and Kazman, *security* is a measure of the systems ability to resist the unauthorized usage while still providing its services to legitimate users[Bass et al., 2003]. Below we have identified the following use cases for assuring the security of the application.

S1 Secure data storage
The application shall store sensitive data secured.

	Values
Source	Developers
Stimulus	Add security measures.
Artifact	System
Environment	Run time
Response	Secure data
Response measure	After positioning the ships, the user is able to change these positions, one at a time.

S2 Authentication
The users shall be authenticated to view sensitive data.

	Values
Source	End User
Stimulus	Requesting to view sensitive data.
Artifact	System
Environment	Run time
Response	Check authentication
Response measure	If authenticated show the appropriate data.

Chapter 7

Design

In this chapter the design process will be explained in detail. We will show how our tool went from sketch to a fully working prototype. Through the different iterations of development, we made different design choices that helped design the application. Before the actual development began, we went through a design process that resulted in a basis for implementation. In this section we will elaborate on how these steps culminated into the final design choice.

7.1 Responsive web design

Responsive web design[Marcotte, 2013c] is a web design approach aimed at developing websites in order to provide an optimal viewing experience on a wide range of devices, from large desktop monitors to smartphones and tablets. This includes easy reading and navigation with minimal resizing, panning, and scrolling.

Any site designed to be responsive, adapts the layout to the viewing environment, along with fluid proportion-based grids and flexible images. An illustration of how responsive web design adapts to different devices can be seen in Figure 7.1.

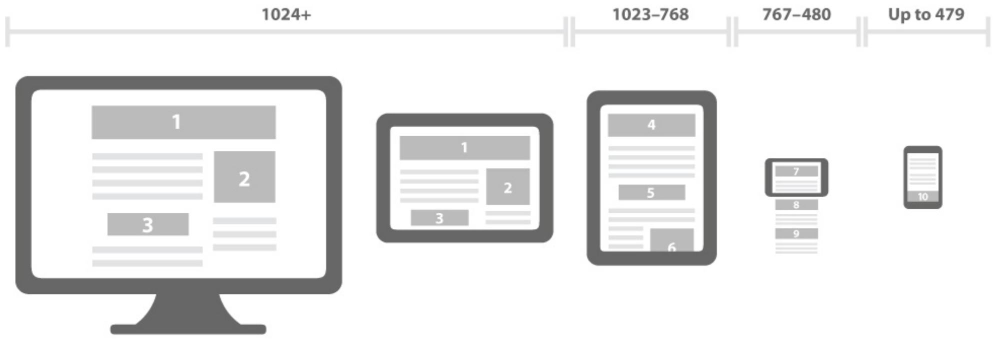


Figure 7.1: Responsive layout, adapting the same content to different viewing experiences [Vinaganda.com, 2013]

7.1.1 Fluid grid

A definition on fluid is : "A fluid is a substance that continually deforms (flows) under an applied shear stress"[Nimesh, 2013]. In a web-design context, fluid will be the design and layout, and shear stress will be the device used to access the content. Regardless of what the device or screen size is, components in fluid designs are going to flow and adapt to the user environment. Fluid grids define a maximum layout size and the grid is divided into columns for easier handling. These grids can then be designed with proportional widths and heights. The fluid grid and it's columns can be seen in Figure 7.1.

The fluid grid[Nimesh, 2013; Marcotte, 2013a] concept states that page elements should be sized in relative units, like percentages or ems, instead of absolute units like pixels or points. Since fluid grids flow naturally based on the dimensions of its parent container-component, specific adjustments for various devices are therefore kept to a minimum. In the fluid grid, flexible images are also sized in relative units, up to 100%, in order to prevent them from stretching outside their containing element[Marcotte, 2013b]. This means all kinds of elements on a web-page can be treated as proportions measured against their container, and not in absolute pixels.

7.2 Mockups

At the very beginning we had an idea of what we wanted to do and how to do it. At the time we had not decided on a platform for our application. Because of this we sketched up some different design choices.

7.2.1 Smartphone App

Our first alternative was creating a native app for smartphones, on iOS or Android. We had experience with creating android applications before, so this was our first thought.

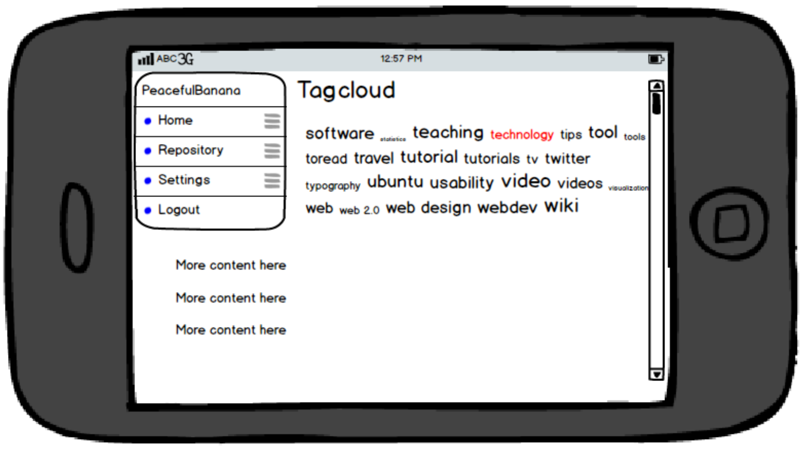

ABC 3G
12:57 PM
PeacefulBanana
Home
Repository
Settings
Logout
More content here
More content here
More content here
Tagcloud
software statistics teaching technology tips tool tools
toread travel tutorial tutorials tv twitter
typography ubuntu usability video videos visualization
web web 2.0 web design webdev wiki

Figure 7.2: Smartphone mockup

7.2.2 Web-Application tool

Our second option was creating our tool as a web-application. Web-applications are platform independent and can be accessed on a wide range of devices, as long as it has a fairly updated browser. The idea was to also make the web application responsive, which would provide an optimized viewing experience allowing for easy reading and navigation with minimal resizing, panning, and scrolling across a wide range of devices (from large monitors to mobile phones
.
Figure 7.3 shows how our first sketch for a web-app looked like.

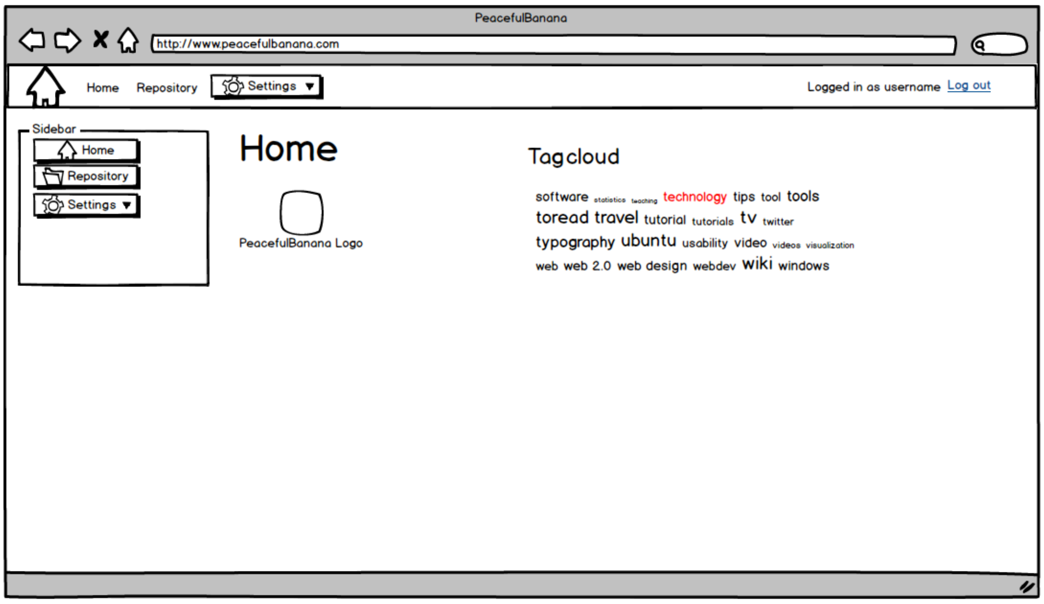

PeacefulBanana
http://www.peacefulbanana.com
Home
Repository
Settings
Logged in as username
Log out
Sidebar
Home
Repository
Settings
Home
PeacefulBanana Logo
Tagcloud
software statistics teaching technology tips tool tools
toread travel tutorial tutorials tv twitter
typography ubuntu usability video videos visualization
web web 2.0 web design webdev wiki windows

Figure 7.3: Web-app mockup

As this was the main design options, we did some research on existing frameworks which might suit our requirements. The tool needed to be responsive and work on a wide range of devices, which made a fluid design suitable.

7.3 Twitter Bootstrap

The Twitter Bootstrap framework[Twitter, 2013] features both the fluid grid and responsive web design requirements stated above. Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter, as a framework to promote consistency in internal tools[Otto, 2013]. Twitter bootstrap is a free powerful front-end framework for faster and easier web development, and it's publicly available to use by anyone.

It contains HTML and CSS based design templates for interface elements like buttons, charts, and navigation. Bootstrap was made to not only look and behave optimally in desktop browsers, but in tablet and smartphone browsers via responsive CSS and HTML5 as well. The framework features responsive grids, components like tabs and dropdowns, JavaScript plugins, typography options and forms.

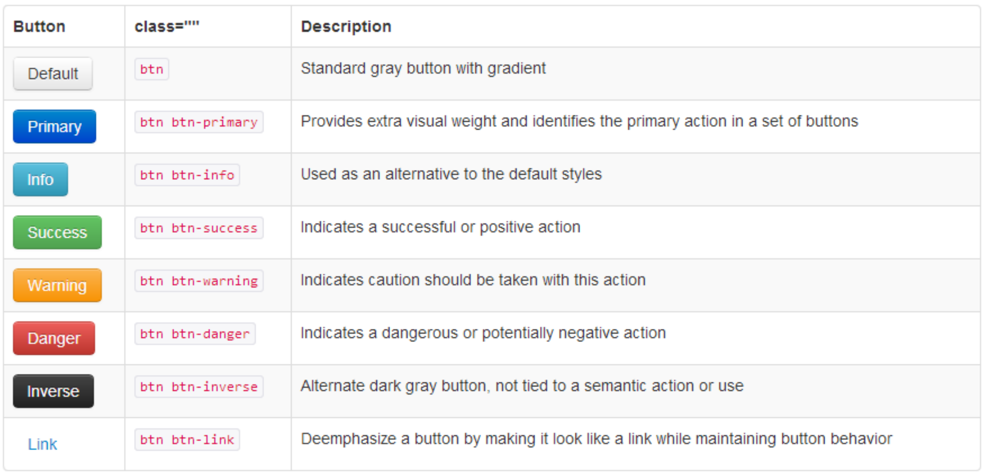
Twitter Bootstrap is the most popular project in GitHub and is used by amongst others NASA(National Aeronautics and Space Administration)[Aeronautics and NASA, 2013] Bootstrap is also modular, which means it is composed of a series of components that together make the toolkit. Developers can then cherry pick the components they need from the Bootstrap toolkit.

7.3.1 Bootstrap grid

Bootstrap ships with the standard 940 pixel wide, grid layout. Developers can choose to use a variable-width layout if they wish. In any case the Bootstrap toolkit has four major variations in order to adapt content and grid width to different resolutions and devices: mobile phones, both portrait and landscape format, tablets and desktop computers with both low and high resolution and wide-screen support.

7.3.2 Bootstrap components

Bootstrap features a set of CSS style sheet, which defines styles for the major HTML elements. These style sheets allow for a platform-independent, multi browser enabled and consistent appearance for text, tables and other elements. Bootstrap also comes with additional commonly used interface components. Such components are buttons (See Figure 7.4, button-groups, buttons with drop-down option, navigation lists, tabs, breadcrumbs, pagination, different alerts and so on.



Button	class=""	Description
Default	`btn`	Standard gray button with gradient
Primary	`btn btn-primary`	Provides extra visual weight and identifies the primary action in a set of buttons
Info	`btn btn-info`	Used as an alternative to the default styles
Success	`btn btn-success`	Indicates a successful or positive action
Warning	`btn btn-warning`	Indicates caution should be taken with this action
Danger	`btn btn-danger`	Indicates a dangerous or potentially negative action
Inverse	`btn btn-inverse`	Alternate dark gray button, not tied to a semantic action or use
Link	`btn btn-link`	Deemphasize a button by making it look like a link while maintaining button behavior

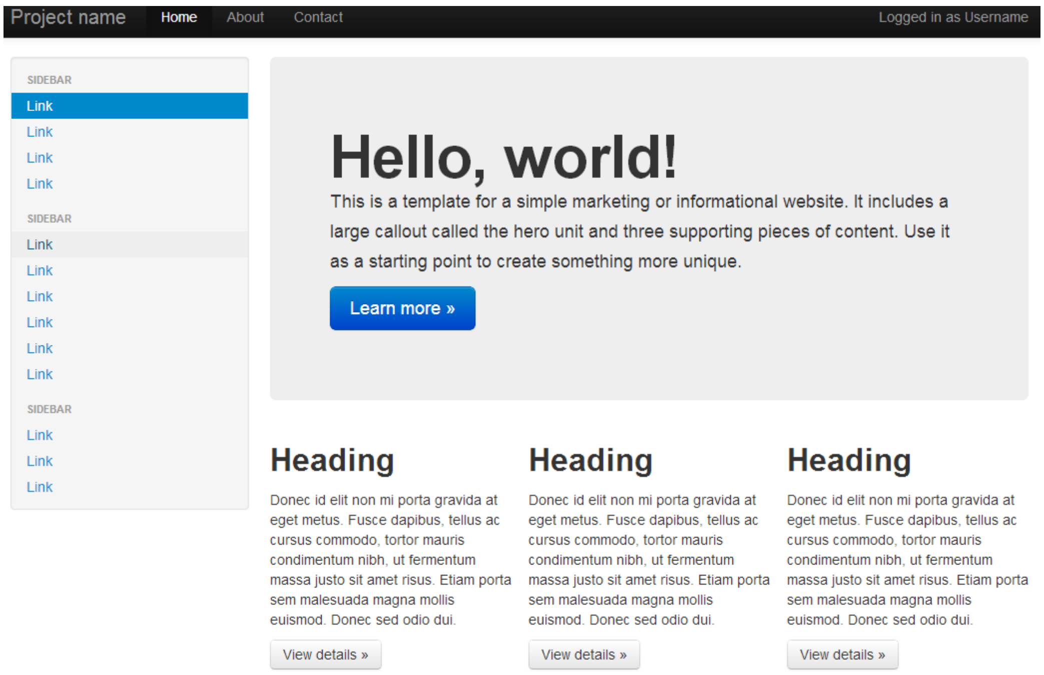
Figure 7.4: Example of Twitter Bootstrap button components [Twitter, 2013]

7.3.3 Icons

Twitter Bootstrap comes with 140 sprite form icons, in both dark gray and white[1]

7.3.4 Design Examples

At the Twitter web-site they feature some examples that can be used as a basis for development. We decided to base our web application design on this layout, as it suited our needs and was very close to our initial mock-up. Figure 7.5 shows the example project we chose for the PeacefulBanana application.



Hello, world!

This is a template for a simple marketing or informational website. It includes a large callout called the hero unit and three supporting pieces of content. Use it as a starting point to create something more unique.

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

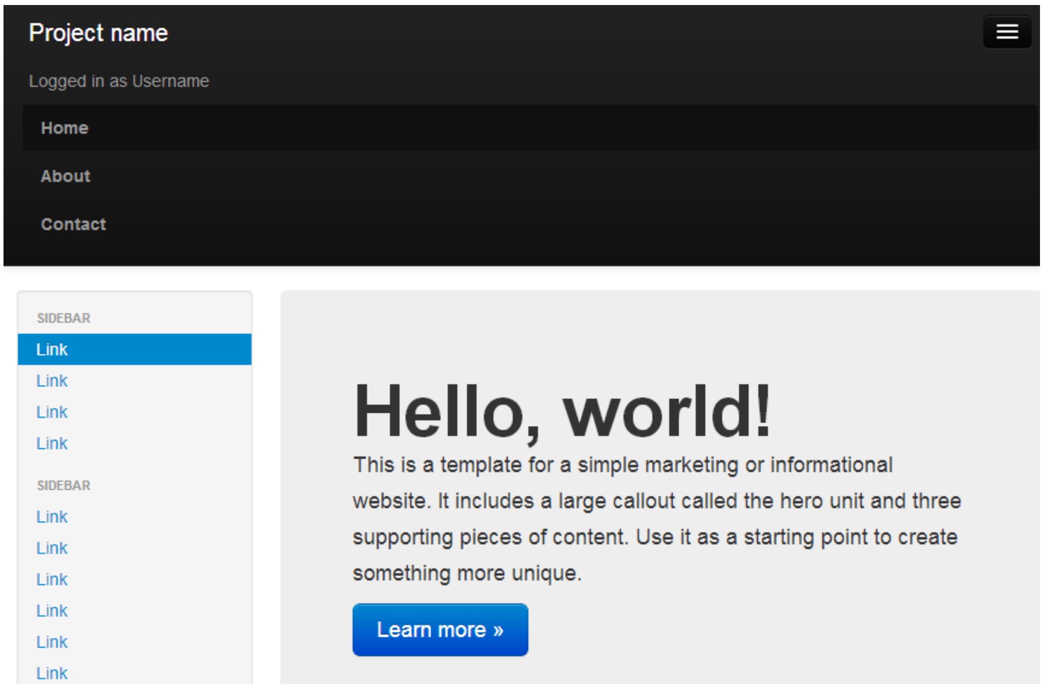
Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

Figure 7.5: Twitter bootstrap fluid layout with header and sidebar [Twitter, 2013]

Visiting the site on a mobile device with a smaller screen, the responsive fluid layout would optimize the page for that device, as shown in Figure 7.6.

[1]Bootstrap icons: `http://twitter.github.com/bootstrap/base-css.html#icons`



Hello, world!

This is a template for a simple marketing or informational website. It includes a large callout called the hero unit and three supporting pieces of content. Use it as a starting point to create something more unique.

Figure 7.6: Twitter Bootstrap fluid layout with responsive menu and content [Twitter, 2013]

Figure 7.7 is how our web-app looked after implementing bootstrap for a fully fluid and responsive web-application:

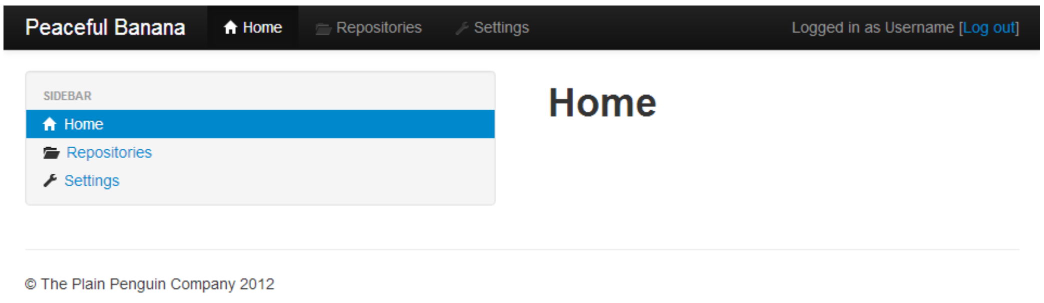


Figure 7.7: PeacefulBanana - Initial setup with Twitter Bootstrap

7.4 PeacefulBanana Design

After setting up and integrating the twitter bootstrap framework, the PeacefulBanana web-application was born, fully responsive and available on multiple platforms. Figure 7.8 shows the final PeacefulBanana design, which emerged from the starting design shown in Figure 7.7:

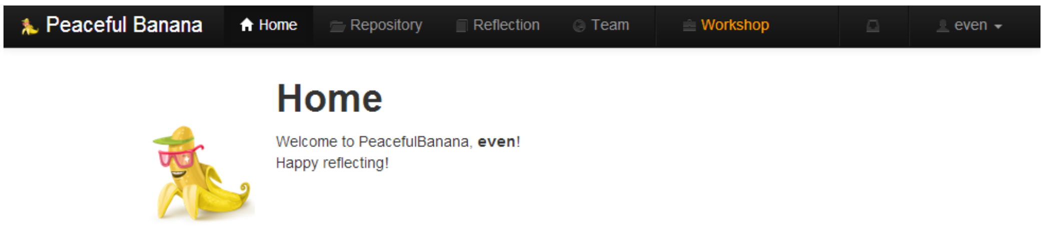


Figure 7.8: PeacefulBanana - Final home screen

The design is based on the Twitter Bootstrap framework and the principles of responsive web-design and a fluid layout. More specifically it is based on the fluid layout example from Figure 7.5[2]. The PeacefulBanana layout features a navigation-menu on top with different tabs, leading to different content. Each of these tabs contain sidebars, which acts as sub-menus. The tool also integrates a user-menu, which is a drop-down with different actions that relates to the user, like settings.

7.4.1 Icons

The PeacefulBanana tool makes extensive use of the icons included in Twitter Bootstrap, in order to further define what action you can expect from a tab. For example a home icon on the *Home* tab, a globe on the *Team* tab and a user icon on the *User dropdown*, which can be seen in Figure 7.9.

[2]`http://twitter.github.com/bootstrap/examples/fluid.html`

Figure 7.9: Example of Twitter Bootstrap/Glyphicons use in PeacefulBanana [Twitter, 2013]

7.4.2 MIRROR CSRL

This section describes the rationale behind the most important functionality implemented to PeacefulBanana, with relation to the Mirror CSRL cycle model. The section ends with a more detailed rationale behind the core functionality implemented. As described in Section 3.1.1 , the MIRROR CSRL model has four stages:

1. Plan and do work
2. Initiate Reflection
3. Conduct Reflection session
4. Apply Outcome

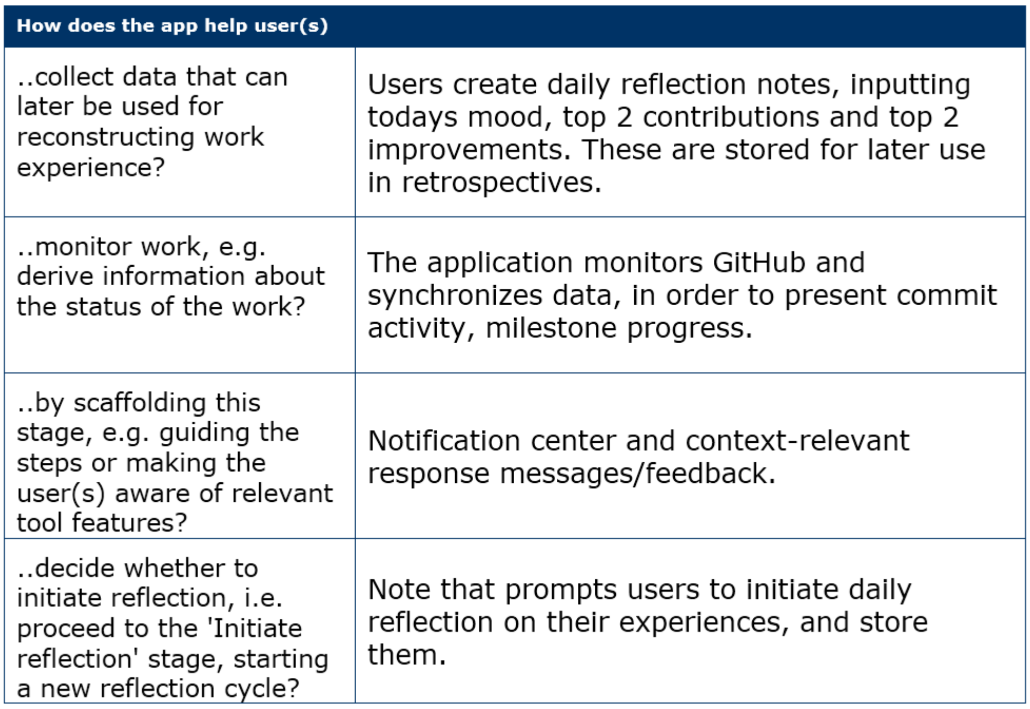
In this thesis, our research mostly focuses on the first two stages of the model, since PeacefulBanana is primarily to be used during work and as preparation for retrospective sessions. PeacefulBanana additionally includes some implementations focused towards the third stage *Conduct reflection session*, as it also can be used during the session.

7.4.3 Plan and do work

Figure 7.10 shows functionality we implemented towards the *Plan and do work* stage, and how our application help user(s) plan and perform their daily work tasks. Functionality in this stage is connected to **Sub research question 1**, which is focused on answering how the data collected from GitHub can be scaffolded in order to promote reflection. By carefully examining what GitHub offers itself, we could identify what they didn't offer and implement this functionality provided it could have a positive effect for promoting reflection for users. The challenge was not only to identify what data to collect, but also scaffold the collected data and present them to the users in a way that triggers reflection.

Adding mood to the *Daily Reflection Note* implementation gave us the opportunity to create a mood-graph based on the team user's mood every day. This representation allows for a quick and easy mood comparison, which may encourage user's to look into why their mood differs opposed to the team's in a certain period of time. Such a discussion might occur at any time, either because a user browses on his own and wants to discuss his findings, or the team browses it together. The rationale of adding notifications to the application was made in order to give users context-relevant feedback and remind them of the daily reflection notes.

This functionality is also connected to **Sub research question 2**, which focuses on how to improve the tendency to reflect, both individually and in teams.

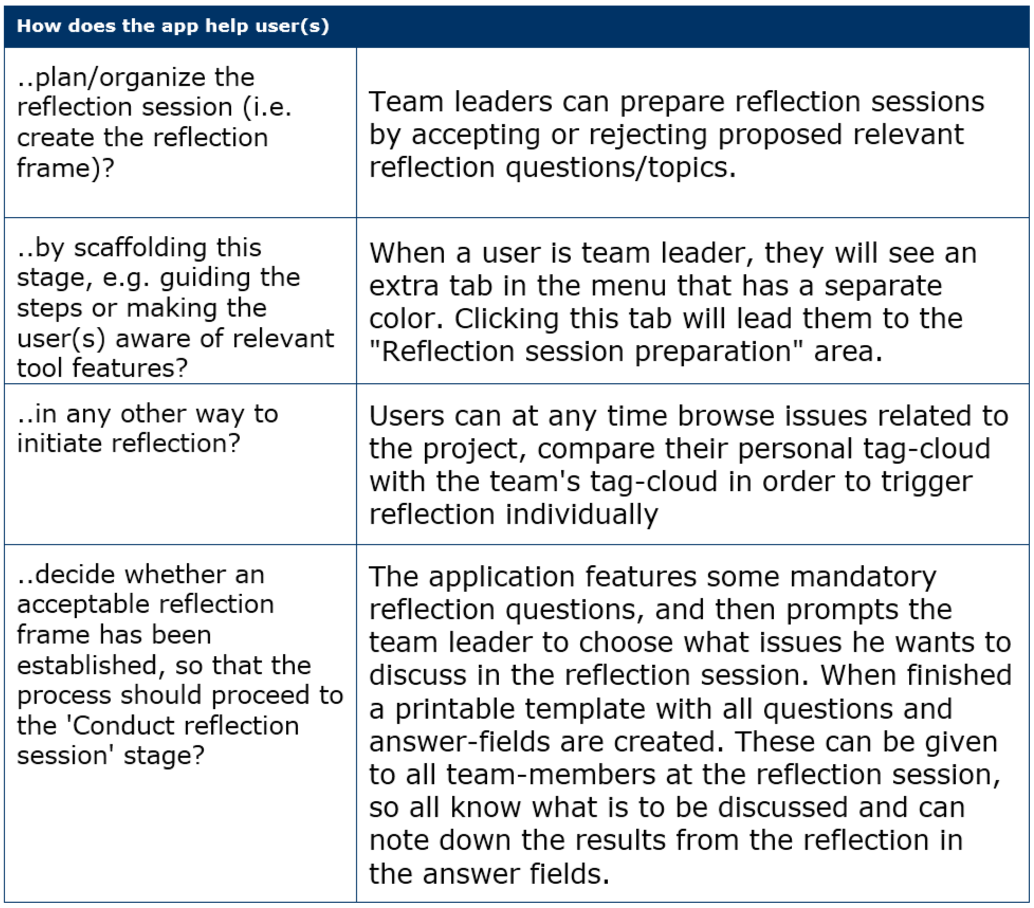


How does the app help user(s)	
..collect data that can later be used for reconstructing work experience?	Users create daily reflection notes, inputting todays mood, top 2 contributions and top 2 improvements. These are stored for later use in retrospectives.
..monitor work, e.g. derive information about the status of the work?	The application monitors GitHub and synchronizes data, in order to present commit activity, milestone progress.
..by scaffolding this stage, e.g. guiding the steps or making the user(s) aware of relevant tool features?	Notification center and context-relevant response messages/feedback.
..decide whether to initiate reflection, i.e. proceed to the 'Initiate reflection' stage, starting a new reflection cycle?	Note that prompts users to initiate daily reflection on their experiences, and store them.

Figure 7.10: Mirror CSRL Cycle - Plan and do work functionality & implementations

7.4.4 Initiate Reflection

Figure 7.11 shows functionality we implemented towards the *Initiate Reflection* stage, and how our application help user(s) initiate reflection. Functionality in this stage is also coupled with **Sub research question 2**. The *Reflection Workshop* functionality is implemented in order for teams to create a frame for their retrospective session. When a workshop has been added, individuals can also prepare by themselves by comparing their own data with the teams, i.e. the personal tag-cloud vs team tag-cloud and the mood-graph.



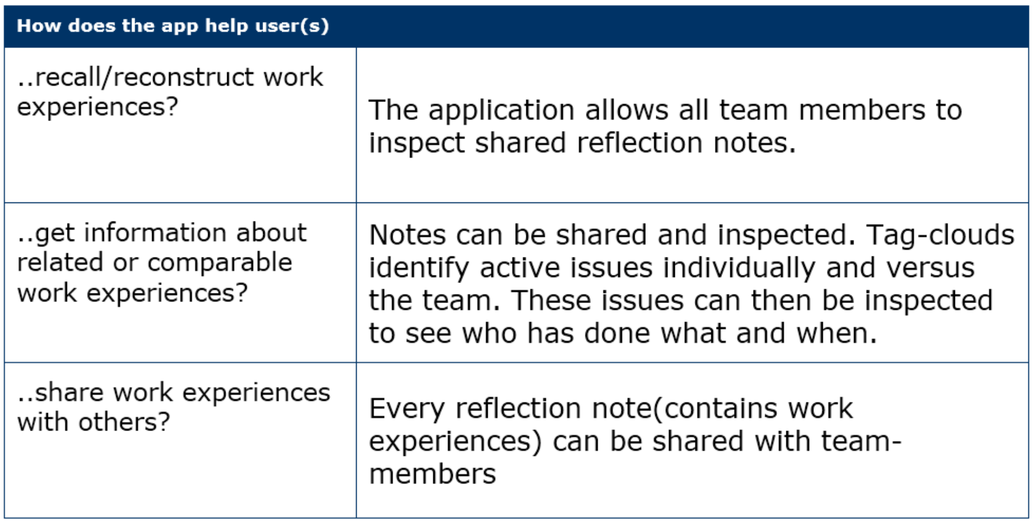
How does the app help user(s)	
..plan/organize the reflection session (i.e. create the reflection frame)?	Team leaders can prepare reflection sessions by accepting or rejecting proposed relevant reflection questions/topics.
..by scaffolding this stage, e.g. guiding the steps or making the user(s) aware of relevant tool features?	When a user is team leader, they will see an extra tab in the menu that has a separate color. Clicking this tab will lead them to the "Reflection session preparation" area.
..in any other way to initiate reflection?	Users can at any time browse issues related to the project, compare their personal tag-cloud with the team's tag-cloud in order to trigger reflection individually
..decide whether an acceptable reflection frame has been established, so that the process should proceed to the 'Conduct reflection session' stage?	The application features some mandatory reflection questions, and then prompts the team leader to choose what issues he wants to discuss in the reflection session. When finished a printable template with all questions and answer-fields are created. These can be given to all team-members at the reflection session, so all know what is to be discussed and can note down the results from the reflection in the answer fields.

Figure 7.11: Mirror CSRL Cycle - Initiate Reflection functionality & implementations

7.4.5 Conduct Reflection Session

Figure 7.12 shows the functionality we implemented towards the *Conduct Reflection Session* stage, and how our application help user(s) perform their retrospectives.

This stage is not something we focused on, since most of the use of PeacefulBanana is doing daily work and using those data to prepare for retrospectives. Still the most notable functionality that is connected to this stage, is the ability to share and inspect reflection notes, meaning that teams can during reflection sessions wish to compare shared notes and create a discussion around these. Similarly teams can inspect repository data, like milestones and issues, and also tag-clouds connected to these. These tag-clouds can identify active issues which the team wishes to discuss during the retrospective.



How does the app help user(s)	
..recall/reconstruct work experiences?	The application allows all team members to inspect shared reflection notes.
..get information about related or comparable work experiences?	Notes can be shared and inspected. Tag-clouds identify active issues individually and versus the team. These issues can then be inspected to see who has done what and when.
..share work experiences with others?	Every reflection note(contains work experiences) can be shared with team-members

Figure 7.12: Mirror CSRL Cycle - Conduct reflection session functionality & implementations

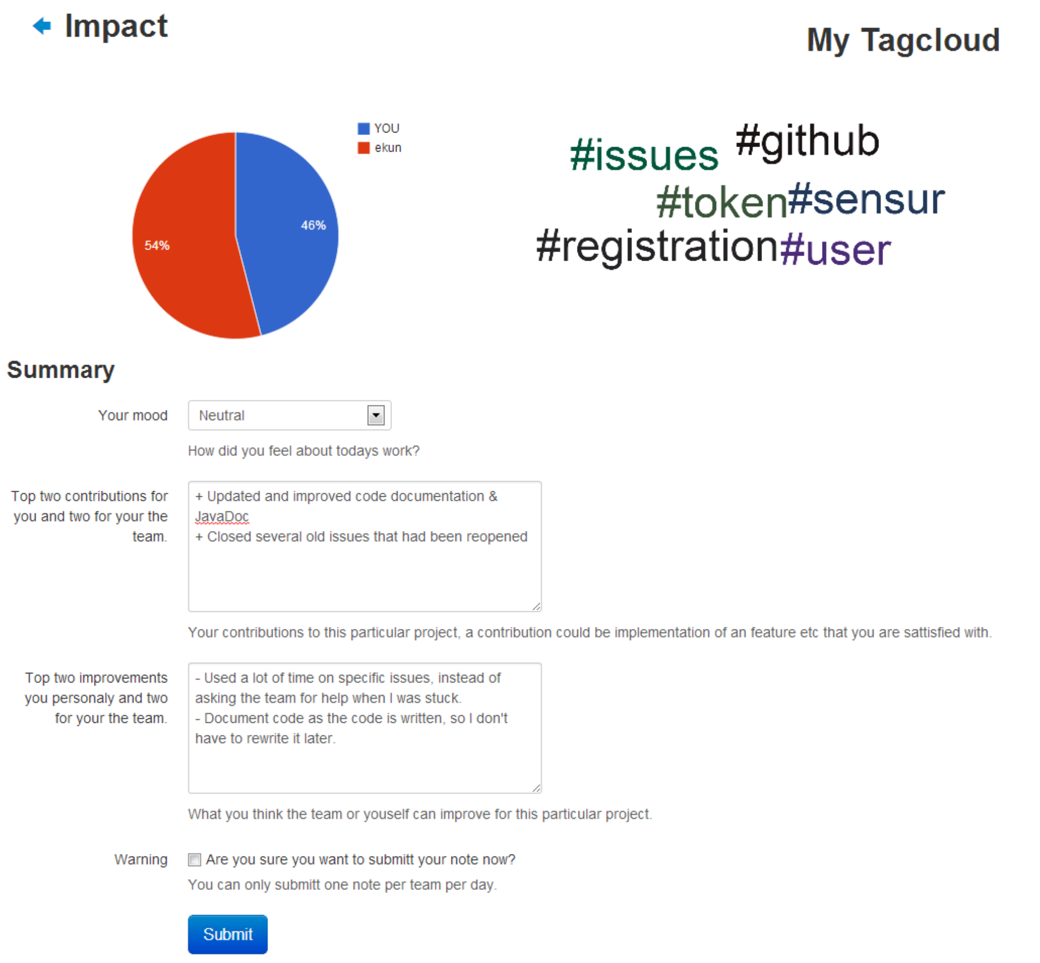
7.4.6 Core Functionality

This section will describe the core functionality in the PeacefulBanana application and the relationship between the application and the CSRL model in section 7.4.2.

Daily reflection note

Scaffolding is the act of creating a skeleton or a frame for the work to be done. PeacefulBanana provides users with the ability to create reflection notes, where they reflect on that day's work and store it for later use. In this context the scaffolding is creating a frame for the reflection to be done, which includes text input and guiding questions. An example of such a reflection note can be seen in Figure 7.13.

In order to encourage users to reflect on their daily work and to avoid generic or non-constructive input, we connected questions to each input field, to act as guidance. For each input, the application provides a question, like *How did you feel about todays work?* for the mood input. Adding these questions helps users think back on their experiences that day and perform a reflection on these.



Summary

Your mood: Neutral

How did you feel about todays work?

Top two contributions for you and two for your the team.

+ Updated and improved code documentation & JavaDoc
+ Closed several old issues that had been reopened

Your contributions to this particular project, a contribution could be implementation of an feature etc that you are satisfied with.

Top two improvements you personaly and two for your the team.

- Used a lot of time on specific issues, instead of asking the team for help when I was stuck.
- Document code as the code is written, so I don't have to rewrite it later.

What you think the team or youself can improve for this particular project.

Warning ☐ Are you sure you want to submitt your note now?

You can only submitt one note per team per day.

Figure 7.13: PeacefulBanana reflection note

Mood graphs

When creating a reflection note, users can connect their current mood, that is how they felt, to the reflection note. By adding the ability to connect mood to notes each day, the tool can present meaningful mood-averages back to the users. These mood averages are used by PeacefulBanana to generate a mood-graph, depicting a user's or a team's mood over a period of time.

By presenting this shared team-average mood, user's get an indication of how the work is perceived by the other group members and the team can create a discussion around certain trends in mood, or even why certain users stand out from the rest of the team's mood.

An example of such a mood graph can be seen in Figure: 7.14

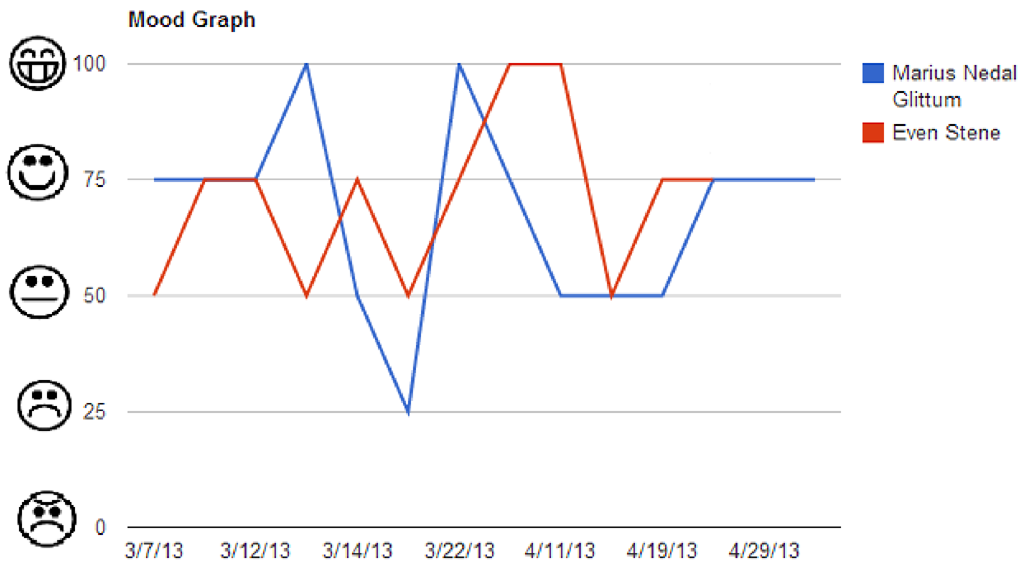


Figure 7.14: PeacefulBanana Team Mood-graph

Tag-clouds

PeacefulBanana also provides the user with individual and team-wide tag-clouds. These tag-clouds are based on the user's or the team's activity in a GitHub project over a period of time. The tag-cloud elements are weighted according to how much the particular *#tag* has been worked with, the more activity the bigger the word. Implementation of the tag-clouds allows user's to see trending events in their own activity trajectory, but also compare it with the rest of the team. This enables the user to see how their work is affecting the team's work, but also gives the team an indication on what has been worked with over different periods of time(i.e. a development iteration).

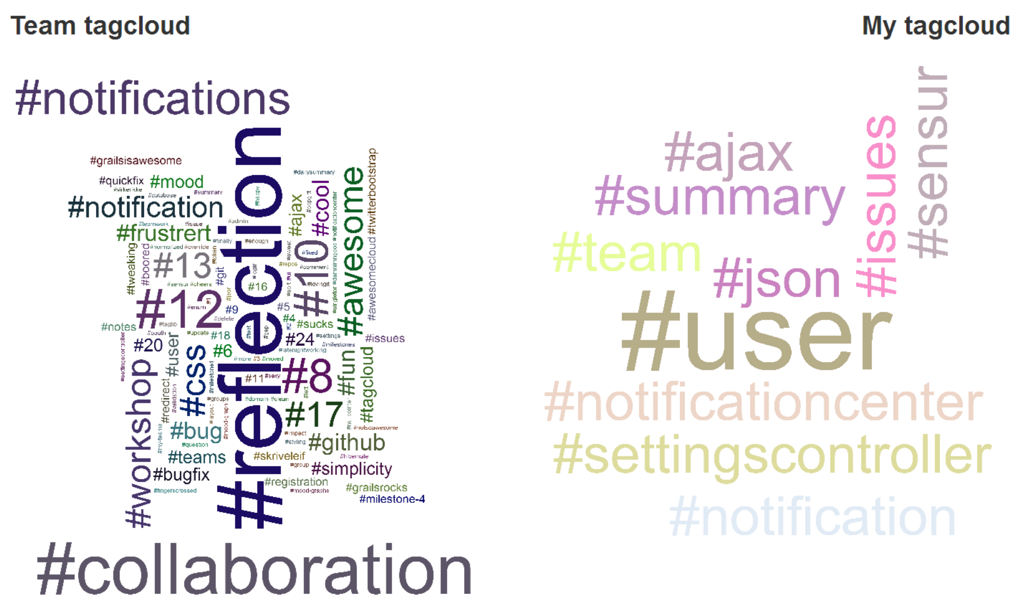
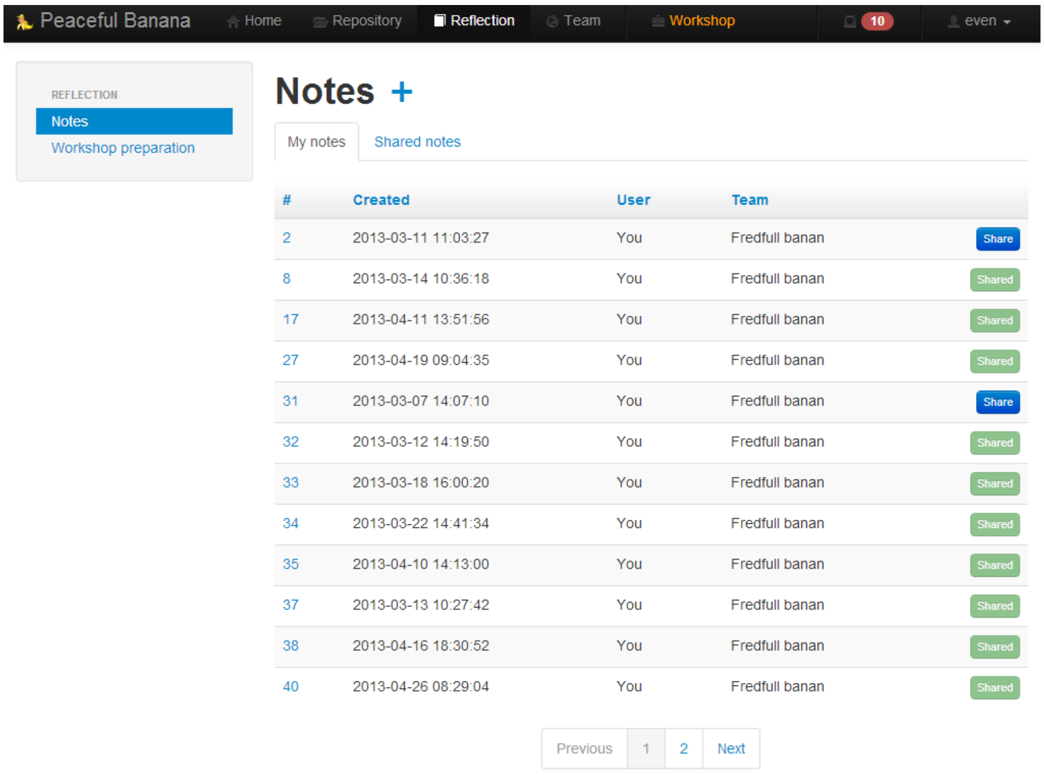


Figure 7.15: PeacefulBanana tag-cloud implementation

Sharing of data

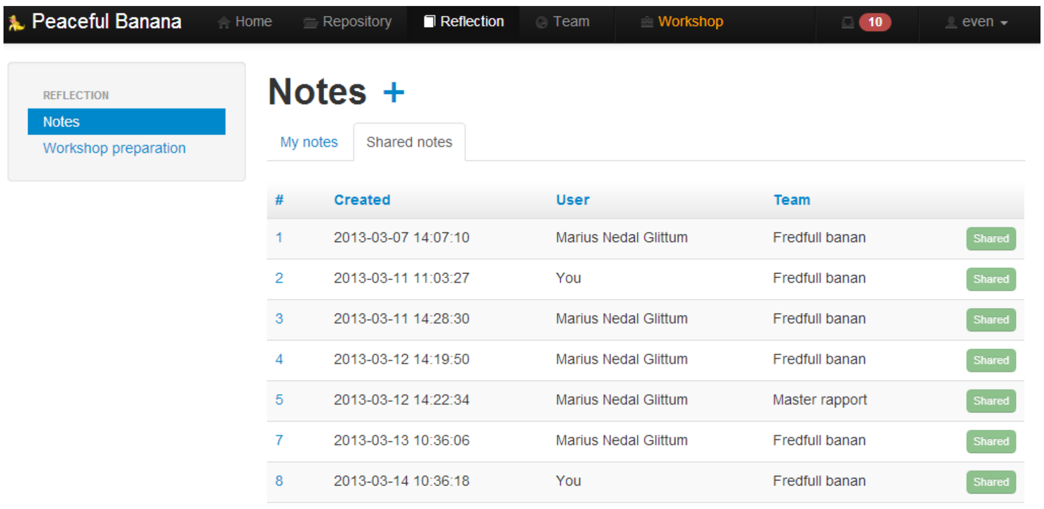
Reflection notes can be shared with a user's team, should they want to. PeacefulBanana collects team-relevant data from GitHub , scaffolds and presents these to users. This data gives the team the ability to see who is working on what, are there any trending issues in the team that can be intercepted and solved and more.

Figure 7.16 shows how PeacefulBanana user's can see their notes and the notes' creation date, owning user, what team that user is on and the share status of the note. The user can also simply filter between their own notes or shared notes by their current team. This is shown in Figure 7.17



#	Created	User	Team	
2	2013-03-11 11:03:27	You	Fredfull banan	Share
8	2013-03-14 10:36:18	You	Fredfull banan	Shared
17	2013-04-11 13:51:56	You	Fredfull banan	Shared
27	2013-04-19 09:04:35	You	Fredfull banan	Shared
31	2013-03-07 14:07:10	You	Fredfull banan	Share
32	2013-03-12 14:19:50	You	Fredfull banan	Shared
33	2013-03-18 16:00:20	You	Fredfull banan	Shared
34	2013-03-22 14:41:34	You	Fredfull banan	Shared
35	2013-04-10 14:13:00	You	Fredfull banan	Shared
37	2013-03-13 10:27:42	You	Fredfull banan	Shared
38	2013-04-16 18:30:52	You	Fredfull banan	Shared
40	2013-04-26 08:29:04	You	Fredfull banan	Shared

Figure 7.16: PeacefulBanana user notes



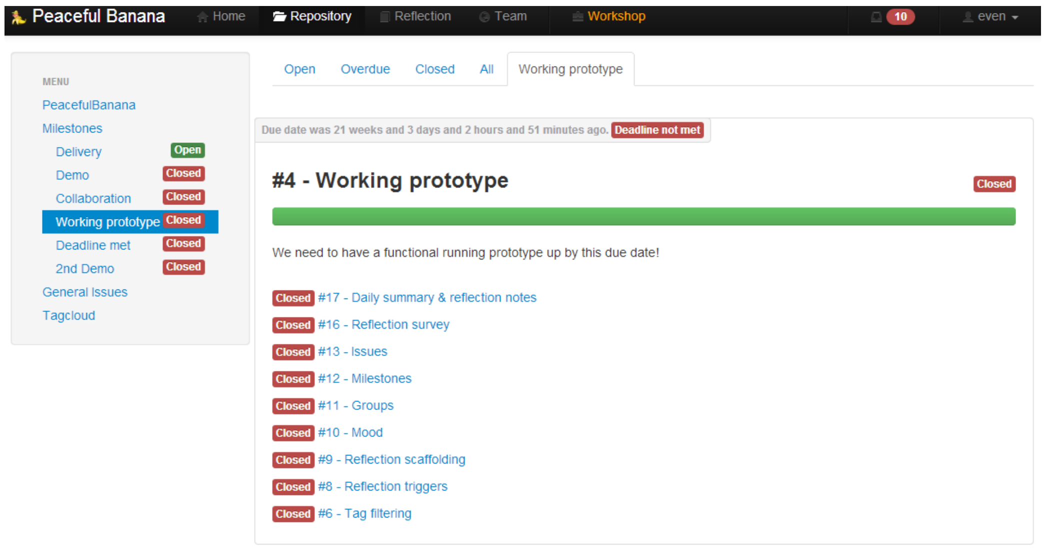
#	Created	User	Team	
1	2013-03-07 14:07:10	Marius Nedal Glittum	Fredfull banan	Shared
2	2013-03-11 11:03:27	You	Fredfull banan	Shared
3	2013-03-11 14:28:30	Marius Nedal Glittum	Fredfull banan	Shared
4	2013-03-12 14:19:50	Marius Nedal Glittum	Fredfull banan	Shared
5	2013-03-12 14:22:34	Marius Nedal Glittum	Master rapport	Shared
7	2013-03-13 10:36:06	Marius Nedal Glittum	Fredfull banan	Shared
8	2013-03-14 10:36:18	You	Fredfull banan	Shared

Figure 7.17: PeacefulBanana shared team notes

Repository

In addition to functionality related to the *Daily Reflection Note* and team retrospective sessions, we also implemented functionality aimed towards individual and collaborative reflection outside of these frames. This includes gaining an overview of the current project, the status of milestones and related issues. The rationale for implementing these functionalities is that we want to keep the required use of PeacefulBanana to a minimum, therefore only the 5-minute daily reflection is required to gain advantages towards the retrospective session. However, if a user or the whole team wants to dive deeper into the project data, they can do so in the *Repository* section. Here users can see the teams commit activity, see the status of milestones and filter them based on this status. Clicking on a specific milestone will show all issues connected to the milestone and a tag-cloud containing that milestone's most used tags. Figure 7.18 shows an example of such a milestone with related issues, and Figure 7.19 shows the tag-cloud connected to this milestone.

This means users or teams can identify certain issues that dominate, and can easily inspect these by clicking on them. Inspecting an issue will present comments, commit references and events concerning that issue. This enables individuals to inspect and revisit experiences, and allows teams to create a discussion around issues or milestones outside of the retrospective sessions.



Due date was 21 weeks and 3 days and 2 hours and 51 minutes ago. Deadline not met

#4 - Working prototype Closed

We need to have a functional running prototype up by this due date!

Closed #17 - Daily summary & reflection notes
Closed #16 - Reflection survey
Closed #13 - Issues
Closed #12 - Milestones
Closed #11 - Groups
Closed #10 - Mood
Closed #9 - Reflection scaffolding
Closed #8 - Reflection triggers
Closed #6 - Tag filtering

Figure 7.18: PeacefulBanana - Repository milestone and issues

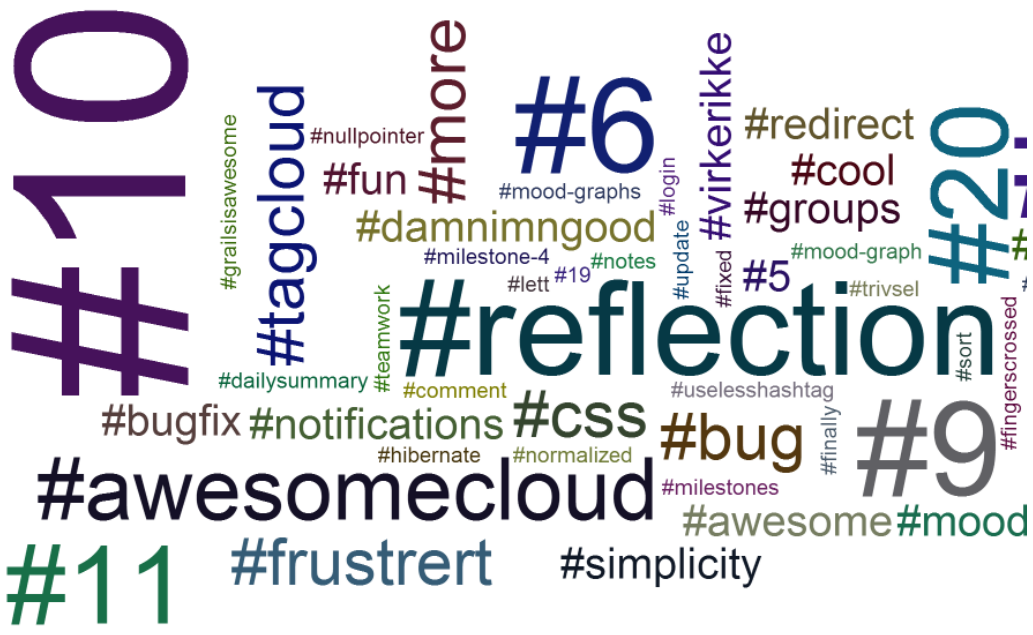


Figure 7.19: PeacefulBanana - Repository milestone tag-cloud

Reflection sessions

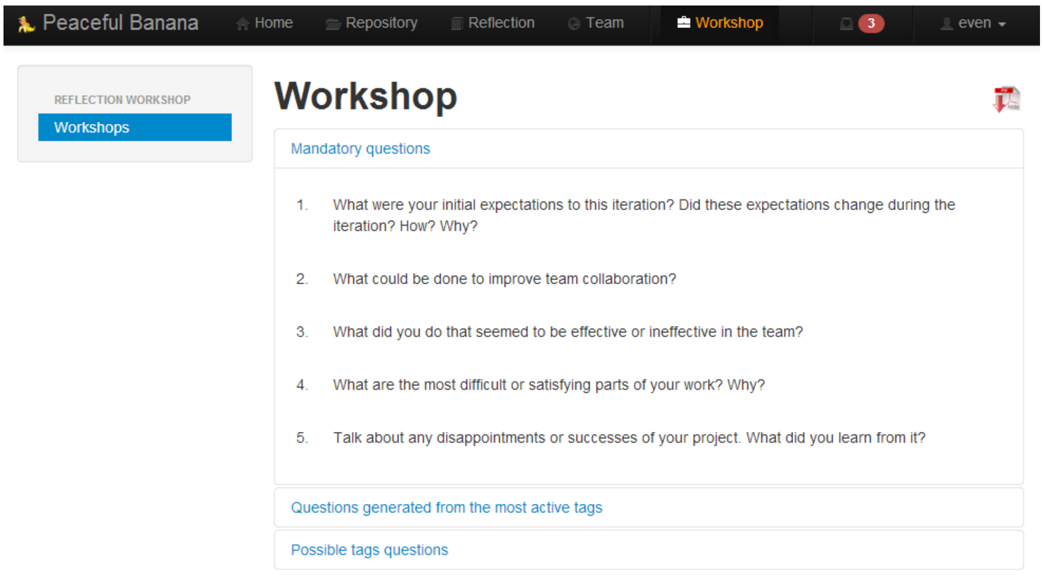
One of the primary functionalities we implemented, in addition to the daily reflection note, was the *Reflection Workshop*. The manager of each team will see a separate *Workshop* tab. The workshop area allows managers to prepare for retrospective sessions, by creating a workshop for that period of time(i.e. an iteration that have lasted for the last two weeks). Figure 7.20 shows an example of such a workshop.

When the workshop has been created, users are met with three accordion headers[3]. The accordions contain some mandatory questions which is some general reflection questions that is relevant to discuss in retrospective sessions, and so these cannot be removed from the workshop. The Peaceful-Banana application also uses gathered data about the most used tags, to generate some proposed questions. The rationale behind this choice is that if the team has had a high activity with a certain tag, it is a high possibility that it's also important to discuss. Examples of these generated questions can be seen in Figure 7.21. Ultimately though, it is the manager of the team that chooses what questions or tags that should be discussed in the

[3]An accordion is a collapsible content panel for presenting information in a limited amount of space

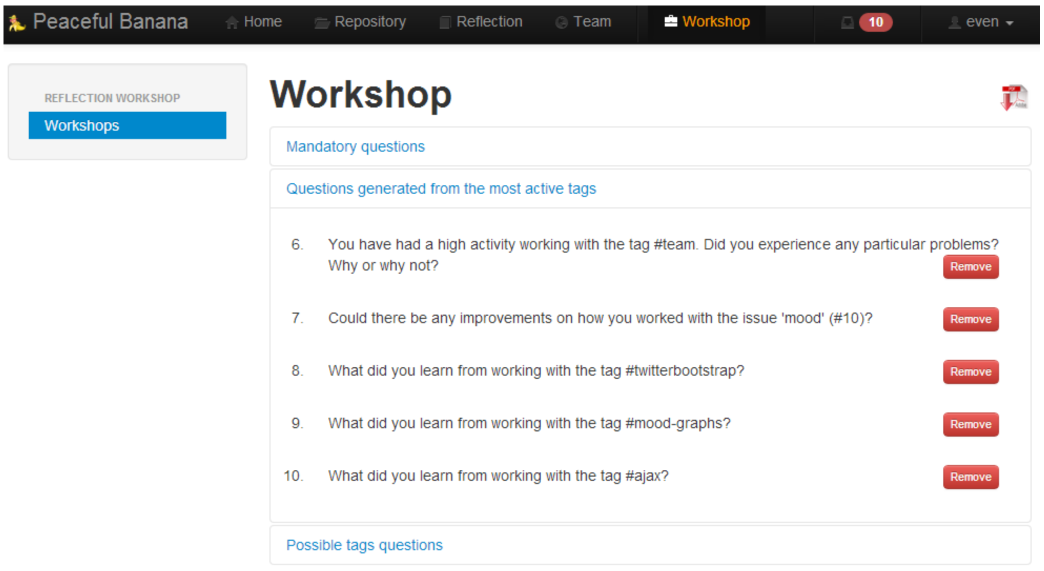
retrospective session. Therefore if the manager feels that one or more of the generated questions are unimportant, they can be removed.

Finally the manager can browse a list of all tags, with the most active at the top, and simply click the tag that should be discussed as part of the retrospective session. If a tag is clicked under the *Possible tags questions* they will be instantly moved to the *Questions generated* section, and vice-versa deleted questions will be moved to the list of possible tag questions. Figure 7.22 shows how the tag list is presented in the application.



1. What were your initial expectations to this iteration? Did these expectations change during the iteration? How? Why?
2. What could be done to improve team collaboration?
3. What did you do that seemed to be effective or ineffective in the team?
4. What are the most difficult or satisfying parts of your work? Why?
5. Talk about any disappointments or successes of your project. What did you learn from it?

Figure 7.20: PeacefulBanana Workshop - Mandatory questions



Workshop

Mandatory questions

Questions generated from the most active tags

6. You have had a high activity working with the tag #team. Did you experience any particular problems? Why or why not? Remove

7. Could there be any improvements on how you worked with the issue 'mood' (#10)? Remove

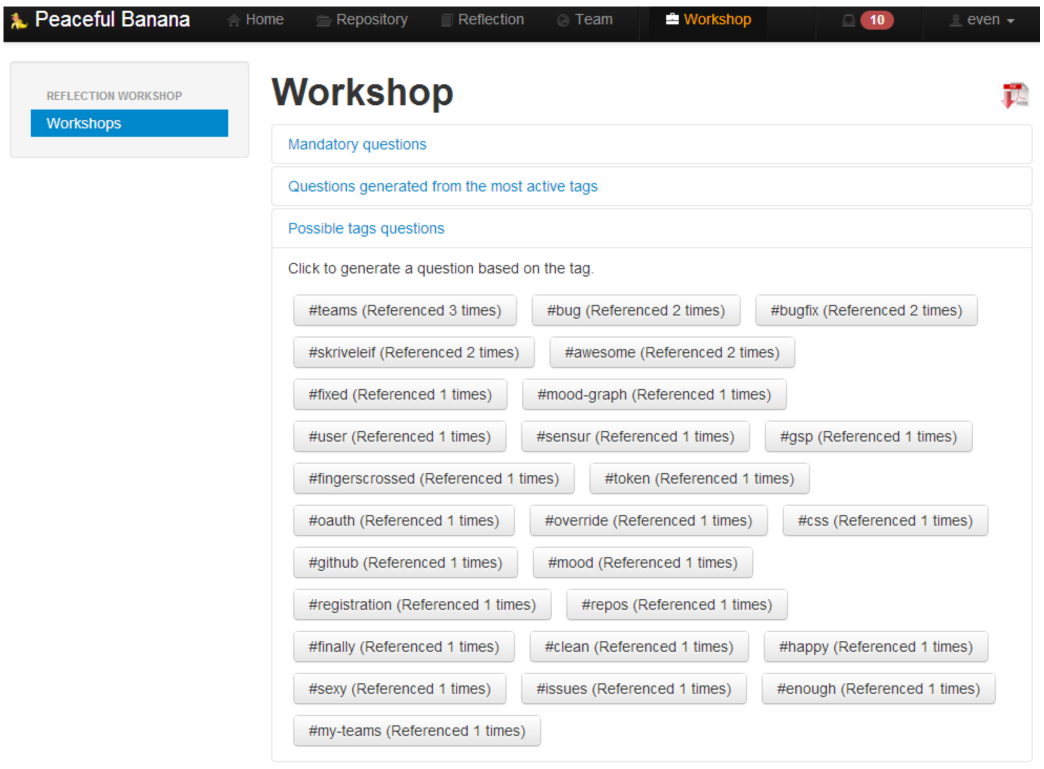
8. What did you learn from working with the tag #twitterbootstrap? Remove

9. What did you learn from working with the tag #mood-graphs? Remove

10. What did you learn from working with the tag #ajax? Remove

Possible tags questions

Figure 7.21: PeacefulBanana Workshop - Generated questions based on active tags



Workshop

Mandatory questions

Questions generated from the most active tags

Possible tags questions

Click to generate a question based on the tag.

#teams (Referenced 3 times) #bug (Referenced 2 times) #bugfix (Referenced 2 times)
#skriveleif (Referenced 2 times) #awesome (Referenced 2 times)
#fixed (Referenced 1 times) #mood-graph (Referenced 1 times)
#user (Referenced 1 times) #sensur (Referenced 1 times) #gsp (Referenced 1 times)
#fingerscrossed (Referenced 1 times) #token (Referenced 1 times)
#oauth (Referenced 1 times) #override (Referenced 1 times) #css (Referenced 1 times)
#github (Referenced 1 times) #mood (Referenced 1 times)
#registration (Referenced 1 times) #repos (Referenced 1 times)
#finally (Referenced 1 times) #clean (Referenced 1 times) #happy (Referenced 1 times)
#sexy (Referenced 1 times) #issues (Referenced 1 times) #enough (Referenced 1 times)
#my-teams (Referenced 1 times)

Figure 7.22: PeacefulBanana Workshop - Possible questions based on tags

Chapter 8

Implementation

This chapter described the PeacefulBanana architecture, the technology we used to implement it and the functionality it provides.

8.1 Application Architecture

As described in Section 6.1.3, the application is implemented with a client-server architecture and all related requirements is described in 6.1.3. Figure 8.1 shows an overview of the system design and the interaction between users, the web-server and GitHub. Each PeacefulBanana user visits the web-application through a web-browser. PeacefulBanana runs on a dedicated server and is able to serve many users at the same time. In the background the PeacefulBanana application communicates with GitHub through the GitHub API (See Section 8.2) and synchronizes project and user-data in real time while users perform their tasks. Users connect their local PeacefulBanana account with their GitHub account in order to gain access to projects. All data collected from GitHub is stored locally on the server in a database (See section 8.2.1).

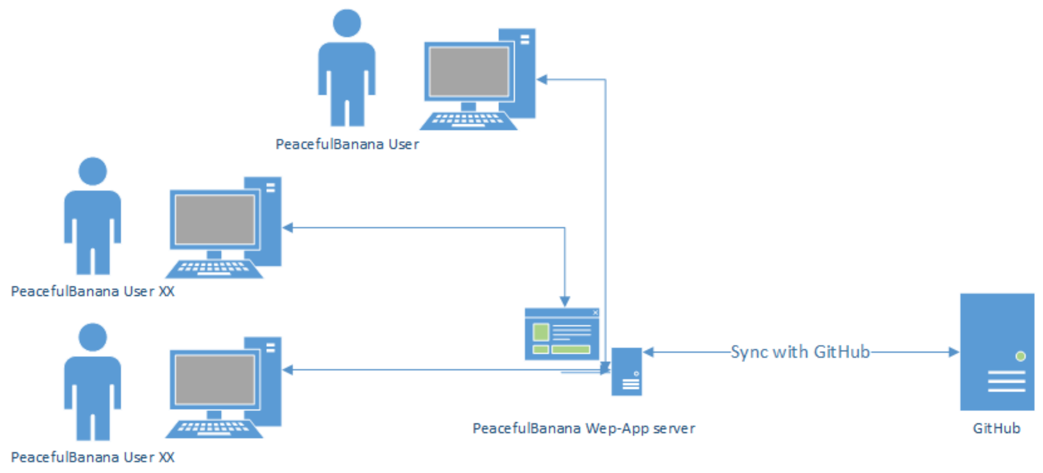


Figure 8.1: Overview of system design.

8.2 Technology

When choosing a framework for the prototype, several alternatives like Spring[1], WebObjects[2] and Play Framework[3] where discussed. Their architecture and our familiarity to the framework was a vital part of the selection. Based on these criteria the server was implemented with Grails a framework for web applications, it is better described in Section 8.2.

GitHub API

GitHub provides developers with the opportunity to use an API[4] which gives them the possibility to communicate with GitHub and retrieve data directly from their database. The identified requirements concerning GitHub is described in Section 6.1.2.

When retrieving data from GitHub we used the provided API as described by their developer-site[5], this enabled us to control the sequence of data and when to ask for what type of data. All communication to GitHub servers

[1]`http://www.springsource.org/`
[2]`htpp://www.apple.com/webobjects/`
[3]`http://www.playframework.com/`
[4]Application programming interface
[5]`http://developer.github.com/v3/`

are asynchronous and while therefore not introduce any performance related issues.

The application needs to authenticate with GitHub in order to , through the use of OAUTH2 tokens. When the user first uses the tool, he will be asked to authenticate with GitHub's authentication page, asking the user to log in and authorize the PeacefulBanana tool. When this is done the tool receives a token it can use on behalf of the user to retrieve data from GitHub. The token itself is not stored, but retrieved in the background when required and bound to the users HTTP-session which expires when the user closes the browser window. This is done for safety reasons cause it would be devastating if these tokes got in the hands of the wrong people, they could for example delete the entire repository.

For gathering data from GitHub, data is transfered as JSON(JavaScript Object Notation), a lightweight data interchange language.

Grails

The PeacefulBanana application was developed with the Grails framework. Grails is an open source web application framework which uses the programming language Groovy(which is built on top of the Java Virtual Machine). When Grails was developed, it's developers aimed to re-use proven technologies such as Hibernate[6] and Spring[7].

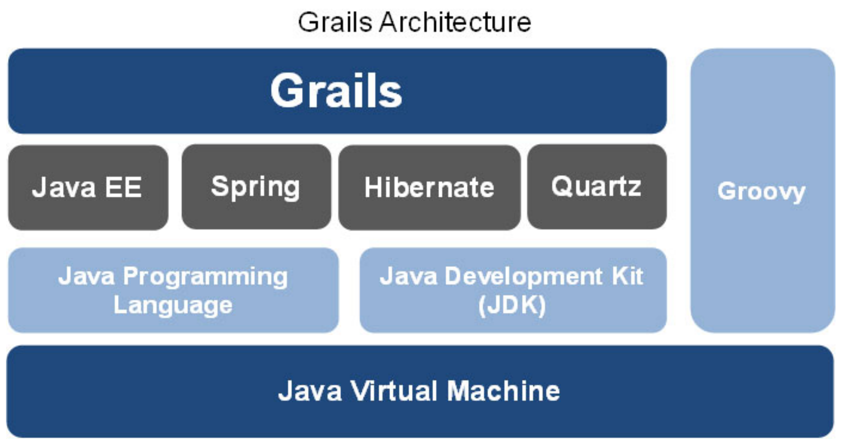


Figure 8.2: Overview of Grails architecture [People10.com, 2013]

[6]http://www.hibernate.org/
[7]http://www.springsource.org/

Architecturally, Grails is designed with the MVC[8] pattern as a basis, this will expose the model[9] in the view[10] and any manipulations to the model is done through a controller which controls that the data is correct input to the fields and what fields can be manipulated. An overview of the paradigm can be viewed in figure 8.3 below.

This pattern makes it easy for the developer to remain in control when creating a user interface and will ensure that the user can not manipulate fields without going through the controller[Reenskaug and Coplien, 2009]. Using MVC increases flexibility and re-use by decoupling the user interface from the model and controller parts. The content of the view must reflect the state of the model, meaning when the model changes it notifies the view, which in turns updates itself.

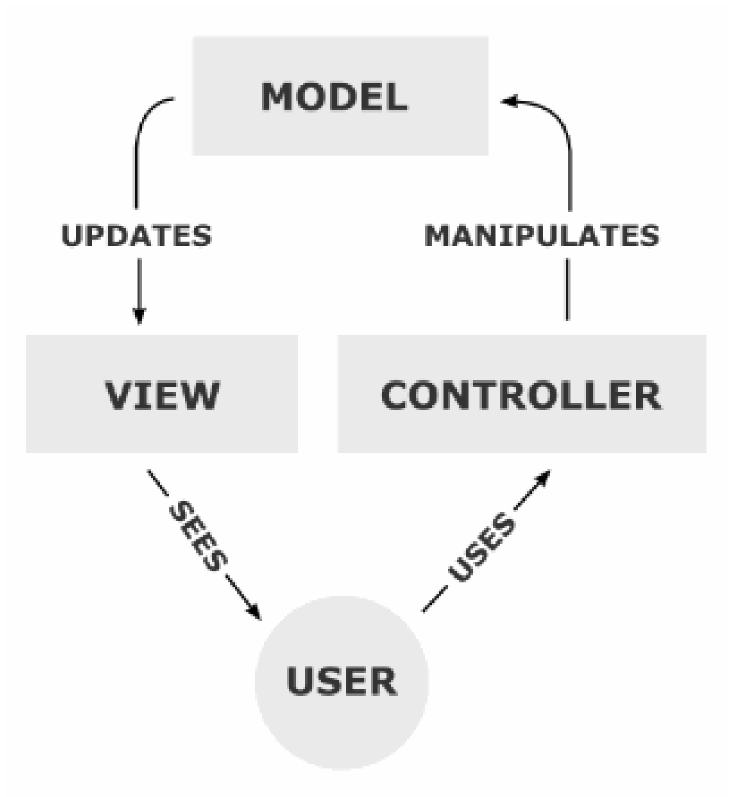


Figure 8.3: Model-view-controller paradigm [Ap and Frey, 2013]

[8]Model View Controller
[9]Data stored about the object.
[10]With the restrictions on what data is viewable for the user.

Spring Security

For authentication and access-control we used the *Spring Security* framework, which is a part of Spring. Spring Security allowed us to create user accounts and connect these within the application. Functionality like changing password, requesting new password when forgotten are examples of what Spring Security provides. Requirements related to authentication and security, is described under *General Requirements* in Section 6.1.1. Requirements related to user accounts, and linking users to teams is described in Section 6.1.2.

jQuery

jQuery was initially released under the MIT license in 2006 to make it easier to select DOM[11] objects and create powerful dynamic web pages and applications.

JQuery provided our application with a way to change objects in real-time without refreshing the whole page, using AJAX[12].

AwesomeCloud

Awesome cloud is a plugin for jQuery for creating a tag cloud. It retrieves data from DOM-objects and renders them as a cloud, where the font-size increases with the occurrence of tags. The tag cloud is then drawn on the HTML5 canvas. We used this to implement our individual and team-wide tag-clouds (See Section 7.4.6).

8.2.1 Server and Database

The PeacefulBanana application was deployed on a Ubuntu server running Apache Tomcat v7. Server related requirements is described in Section 6.1.3. During development we ran the application on a H2[13] in-memory database. However the need for persistent storage arose, so we migrated to a MySQL database in order for the data to stay unchanged when we had to roll out

[11]Document-Object-Model

[12]Asynchronous JavaScript and XML - AJAX is a way to change the contents DOM objects asynchronous from JavaScript.

[13]`http://www.h2database.com/`

updates to the tool. This choice was essential for the tool to function as described in the requirements, but the H2 database was more convenient and faster to debug during development.

MySQL

MySQL[14] is the world's largest open source relational database management system and can be used for a variety of applications. It is most commonly used with Web applications and works very well with dynamic web-pages, meaning that the content of each page is generated based on data loaded from a database as the page loads.

Database description

The PeacefulBanana GitHub specific domain-classes can be seen in Figure 8.4, and shows the relationship between the different data retrieved through the GitHub API. All *User specific* domain-classes can be seen in Figure 8.5.

A detailed walkthrough with data type categorization and description can be seen in Appendix B

[14]MySQL: `http://www.mysql.com/`

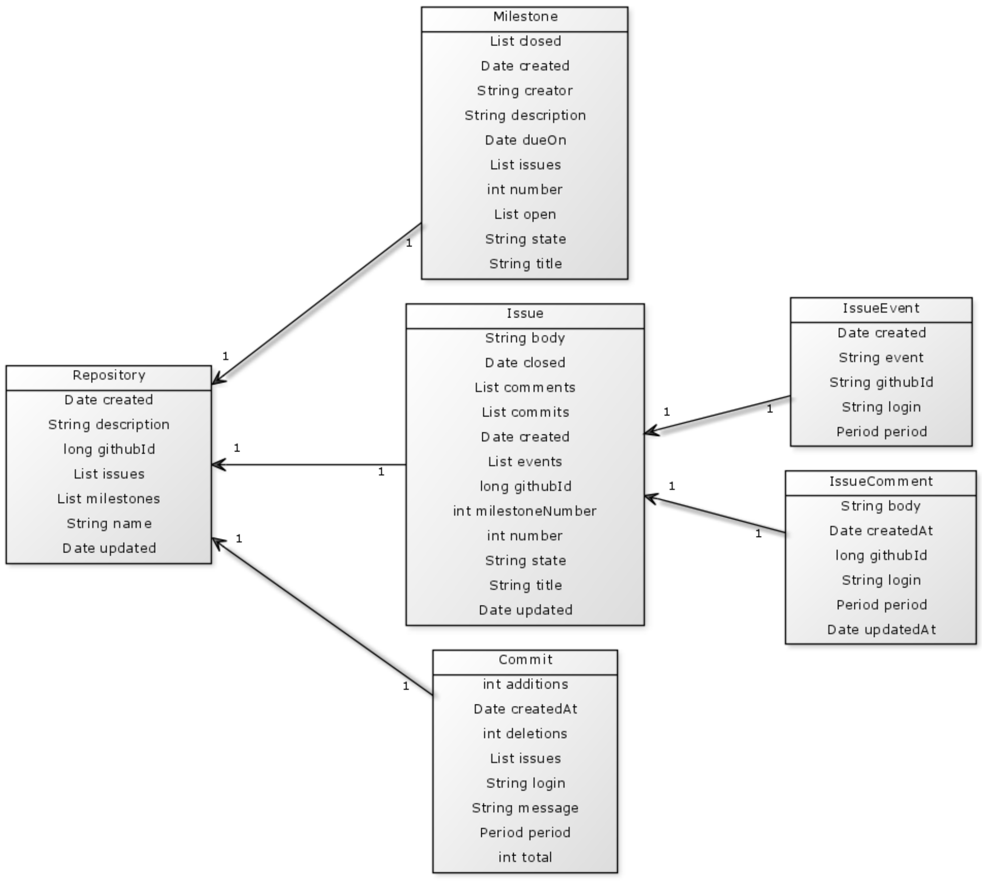


Figure 8.4: GitHub domain classes

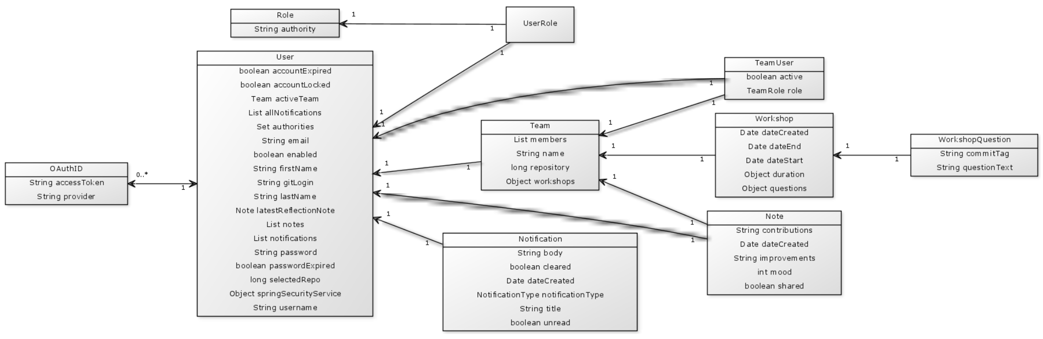


Figure 8.5: User domain classes

Chapter 9

Evaluation

In this chapter we will describe how we evaluated the PeacefulBanana tool, and how it fulfills the requirements identified in Chapter 6.

9.1 Usability Evaluation

In this section we will describe the results and observations made in the usability test. Before we conducted the usability test we created a usability test plan: Appendix C, where all the different parts of the usability test is described in detail.

The usability test was conducted with four participants, recruited from a software development project course at NTNU(IT2901)[1]. The application was evaluated in it's production stage, using the most-recent version at the time. User-interaction with the PeacefulBanana application was done through an Internet Browser(Google Chrome[2]). The users were recruited , and the test was ran with the latest production version of the application.

Users answered an entrance questionnaire, in order to collect demographic information. During the usability test we took notes of the user's problems and concerns. When the test was completed, participants could comment with suggestions on improvement of the application or design. Participants were given a *System Usability Scale* form to answer[Brooke, 1996], which consisted of 10 questions designed to measure user satisfaction.

[1]`http://www.ntnu.edu/studies/courses/IT2901`
[2]`http://www.google.com/intl/no/chrome/browser/`

9.1.1 Context

The usability test simulated the two scenarios identified in Section 5.4 and was set in the context of these. We conducted the usability test in order to answer several important questions, regarding these scenarios:

- Is the application easy to use, and can users achieve their goals in a timely manner?
- Identify the relationship users have with the aspect of reflection and sharing personal experiences.
- Does the tool present data in a way that triggers reflection for the user?

Feedback from the usability test was used to further aid design and help identify problem areas that might cause problems for potential users. Participants were also familiar with reflection, so we hoped to collect valuable input regarding this concept.

9.1.2 Participants

As mentioned we had four participants in the usability test. Typically, three to five test participants is the optimal number for most usability studies[Nielsen and Landauer, 1993]. As the PeacefulBanana tool is intended to be used with developers with a computer-science background, participants were master students on the Computer Science field at NTNU. Participants had experience with usability testing and also experience with the notion of reflection, from earlier projects using an agile methodology process. Responsibilities of participants were to attempt to complete a set of representative task scenarios presented to them in as efficient and timely a manner as possible, and to provide feedback regarding the usability and acceptability of the application. The participants were directed to provide honest opinions regarding the usability of the application.

9.1.3 Procedure

The usability test took place in a private room at the university. A computer with the PeacefulBanana web application was used in a typical working environment. The participants interaction with the application was monitored by the facilitator seated in the same room. The facilitator briefed the participants on the web application and instructed the participants that we are evaluating the application, rather than evaluating the participant. Participants signed an informed consent that acknowledges: *the participation is voluntary, that participation can cease at any time, and that their privacy of identification will be kept safe.* Consent form can be seen in Appendix D.

The facilitator explained that the amount of time taken to complete the test task will be measured and that exploratory behavior outside the task flow should not occur until after task completion. At the start of each task, the participant read aloud the task description from the printed copy and began the task, and time-on-task measurement began simultaneously. The facilitator instructed the participant to *'think aloud'* so that the facilitator could observe and take notes of user behavior and user comments. After all task scenarios are attempted, the participant completed the post-test satisfaction questionnaire derived from the SUS scale[3].

9.1.4 Roles

For our usability test we had two roles, in addition to the test participants:

Facilitator

- Provides overview of study to participants.
- Defines usability and purpose of usability testing.
- Responds to participant's requests for assistance.

Test Observer

- Silent observer
- Takes notes of identified problems, concerns, coding bugs, and procedural errors.
- Serve as note takers.

[3]System Usability Scale

9.1.5 Ethics

All persons involved with the usability test were required to adhere to the following ethical guidelines:

- The performance of any test participant must not be individually attributable. Individual participant's name should not be used in reference outside the testing session.
- A description of the participant's performance should not be reported.

9.1.6 Usability Tasks

The usability tasks were derived from our scenarios, described in Section 5.4. Due to the short time for which each participant was available, the tasks used were the most common and relatively complex of available functionality. The tasks were identical for all participants in the study. The application was tested in a development environment and databases were populated during use, and not pre-populated. This ensured a similar experience as to what the users would get when they first use PeacefulBanana in a real-life setting. The web application was run on a local computer, and not on a dedicated server as it will when deployed in production. This and the possible extra overhead from running the application in development mode, may have an impact on performance slightly in a negative way.

Task context

PeacefulBanana is a tool intended to promote reflection and allow for revisiting and learning from previous experiences. PeacefulBanana integrates with and collects data from the version-control system GitHub.

Scenario 1 tasks:

Here are the tasks participants were to solve related to Scenario 1:

- Task 1: You start the application for the first time, and want to login, link your account with Github and set an active repository.
- Task 2:
  - Task 2.1: View your notifications.

  - Task 2.2: Find the *"Congratulations"* notification and archive it. Find the archive and see if the notification was indeed archived

- Task 3:
  - Task 3.1: Find the *"Reminder: Daily Reflection"* note and perform the daily summary.
  - Task 3.2: Find a daily summary note and share it. Verify that is has indeed been shared.
  - Task 3.3: Find your mood graph

Scenario 2 tasks:

Here are the tasks participants were to solve related to Scenario 2:

- Task 4:
  - Task 4.1: Create a team with the name *"Tuttifrutti"* and your previously chosen repository.
  - Task 4.2: Find your created team and set it to active.
  - Task 4.3: Identify the members on your team and their role.
- Task 5:
  - Task 5.1: Find all your repositories' milestones.
  - Task 5.2: Identify your overdue milestones.
  - Task 5.3: Find your repositories issues.
  - Task 5.4: Find issue #17 . Find the status of this issue, when it was opened and when it was closed.
- Task 6:
  - Task 6.1: Generate a tagcloud for your current chosen repository.
  - Task 6.2: Identify the most used tag for your team and yourself.
  - Task 6.3: Find the commit impact for your repository.

9.1.7 Usability Metrics

Usability metrics refers to user performance measured against specific performance goals necessary to satisfy usability requirements. Scenario completion success rates, error rates, and subjective evaluations was collected, additionally Time-to-completion/Time-on-task was also collected.

Task Completion

Each task requires that the participant obtains or inputs specific data that would be used in course of a typical task. The task is noted as *completed* when the participant indicates the task's goal has been obtained (whether successfully or unsuccessfully). If a participant requires assistance in order to achieve a correct output then the task will be noted as a critical error and the overall completion rate for the task will be affected.

Completion Rate

A completion rate of 100% is the goal for each task in this usability test.
Completion rate is the percentage of test participants who successfully complete the task without critical errors, an *output* that is correct. If a participant requires assistance in order to achieve a correct output then the task will be scored as a critical error and the overall completion rate for the task will be affected.

Critical Errors

A critical error is an error that results in an incorrect or incomplete outcome. Participants may or may not be aware that the task goal is incorrect or incomplete. In general, critical errors are unresolved errors during the process of completing the task or errors that produce an incorrect outcome.

Non-Critical Errors

Non-critical errors are errors that are recovered from by the participant or, if not detected, do not result in processing problems or unexpected results. Although non-critical errors can be undetected by the participant, when they are detected they are often frustrating to the participant.

Subjective Evaluations

Opinions of participators regarding specific tasks, time to perform each task, features, and functionality was collected. At the end of the usability test, participants rated their satisfaction with the overall system. Combined with the interview/debriefing session.

Task Completion Time(time on task)

The time to complete a task is referred to as *"time on task"*. It is measured from the time the person begins the task to the time the participant indicates completion.

9.1.8 General Usability Goals

The general goals of usability testing the PeacefulBanana application included establishing a baseline of user performance, validating user performance measures, and identifying potential design issues that needed to be addressed in order to improve efficiency, usability and end-user satisfaction. The general usability test objectives were:

- Identify possible problems or breakdowns in the design[Wright and Monk, 1989] early on in the design process. Sources of such breakdowns may include:
  - Navigation errors – failure to locate functions, excessive actions to complete a function, failure to follow recommended screen flow.
  - Presentation errors – failure to locate and properly act upon desired information in screens, selection errors due to labeling ambiguities.
  - Control usage problems – improper tool bar or entry field usage.
- Exercise the PeacefulBanana application under controlled test conditions with representative users, which here are individuals with a background in Computer-Science. Data will be used to assess whether usability goals regarding an effective, efficient, and well-received user interface have been achieved.
- Establish baseline user performance and user-satisfaction levels of the user interface for future usability evaluations.

Typical general problems identified in this usability test would be text representations or the placement of design elements, that are not intuitive for the user during use. It would be a concern if the user can't figure out how to use certain features of the application. Identifying these problems as early as possible will lead to a better end result.

Secondly an objective of the usability test was to identify how users act and think about their daily experiences, how they react to the notion of reflecting on them and if sharing their private thoughts is a problem.

9.1.9 Problem Severity

In order to analyze collected data from the usability test, identified issues were classified by issue severity. This issue severity is a combination of the impact of the issue and the frequency of users experiencing the issue during the test.

Problem Severity Classification

- High severity: High impact problems that often prevent a user from correctly completing a task. Reward for resolution is reduced redevelopment costs.
- Medium severity: Either moderate problems with low frequency or low problems with moderate frequency; these are minor annoyance problems faced by a number of participants. Reward for resolution is typically exhibited in reduced time on task and increased data
- Low severity: Low impact problems faced by few participants; there is low risk to not resolving these problems. Reward for resolution is typically exhibited in increased user satisfaction.

Pilot Test

After finalizing the usability test plan, seen in Appendix C, a pilot test was conducted prior to the usability-test[US.gov, 2013]. The pilot test allows for an evaluation of the test plan itself and the questionnaires before doing the actual usability test. This means the pilot test is a *"test of the test"*, where the goal is to evaluate and verify that the test itself is well-formulated.We chose a fellow student as our pilot-tester, in order to check whether the test

script was clear, that the tasks were appropriately difficult, and that the data collected can be meaningfully analyzed. It also allows the "tester" to practice the execution and guidance, before actually performing the tests.
In the pilot test for PeacefulBanana, all of the aspects above were evaluated and a few tweaks were made to the tests, making it more streamlined. Also a few, smaller bugs in the application were discovered and fixed. The test introduction was rewritten, since the pilot-tester showed some confusion in a few of the tasks.

9.1.10 Usability Test Results

We conducted an on site usability test using a production version of PeacefulBanana, located on the test administrators local server. One tester took notes of comments, facial expressions and navigation choices. The administrator acted as guidance during the test. The sessions captured each participants navigational choices, task completion rates, comments, overall satisfaction ratings, questions and feedback. The usability test was conducted in a private lab-room at NTNU on November 10th. The purpose of the test was to assess the usability of the web application design, information flow, information architecture and the effects of sharing personal reflection notes. The findings acted as valuable feedback to our delivery cycle, and were used for improving the design of the application. Four participants attended the test. Each individual session lasted approximately twenty minutes. This section contains the participant feedback, satisfactions ratings, task completion rates, ease or difficulty of completion ratings, time on task, errors, and recommendations for improvements.

Task Completion Success Rate

This section presents the task completion rates for our two scenarios. Figure 9.1 shows the completion rate table for Scenario 1, and Figure 9.2 shows the completion rate table for Scenario 2.

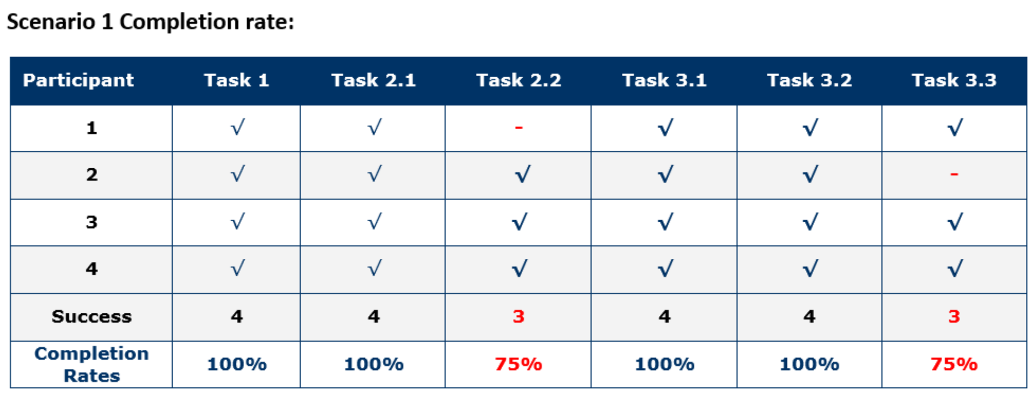
Scenario 1 completion rates:

All participants successfully completed (100% Completion rate):

- Task 1 - Start the application and set active repository.

- Task 2.1 - View notifications.
- Task 3.1 - Find the *Reminder: Daily reflection* note and perform the daily summary.
- Task 3.2 - Find and share the daily reflection note.

For Task 2.2(Find and archive congratulations notification) and Task 3.3(Find mood graph), 3 out of 4 participants completed the tasks(75% Completion rate).



Scenario 1 Completion rate:

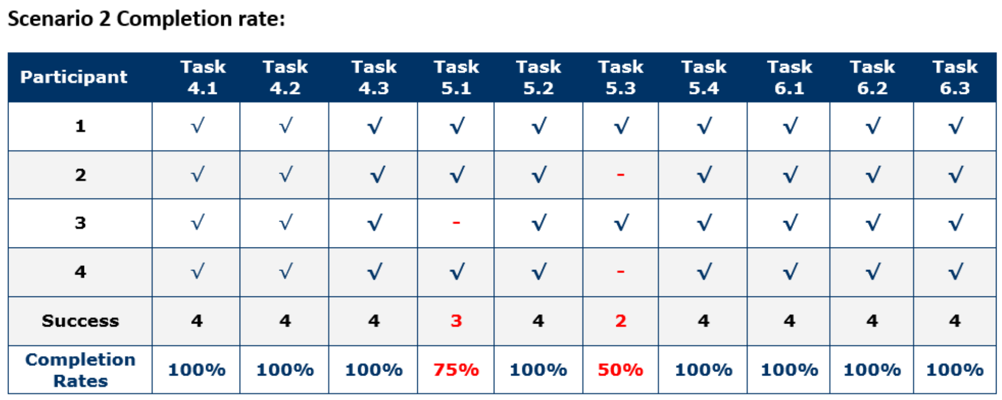
Participant	Task 1	Task 2.1	Task 2.2	Task 3.1	Task 3.2	Task 3.3
1	√	√	-	√	√	√
2	√	√	√	√	√	-
3	√	√	√	√	√	√
4	√	√	√	√	√	√
Success	4	4	3	4	4	3
Completion Rates	100%	100%	75%	100%	100%	75%

Figure 9.1: Scenario 1 Completion Rate

Scenario 2 completion rates: All participants successfully completed (100% Completion rate):

- Task 4.1 - Create a team.
- Task 4.2 - Set active team.
- Task 4.3 - Identify team members.
- Task 5.2 - Identify overdue milestones.
- Task 5.4 - Find issue #17
- Task 6.1 - Generate tag cloud
- Task 6.2 - Identify most used personal tags and team tags
- Task 6.3 - Find commit impact

3 out of 4 participants(75%) successfully completed Task 5.1(Find all milestones for your repository), while 2 out 4(50%) successfully completed Task 5.3 (Find your repositories issues).



Scenario 2 Completion rate:

Participant	Task 4.1	Task 4.2	Task 4.3	Task 5.1	Task 5.2	Task 5.3	Task 5.4	Task 6.1	Task 6.2	Task 6.3
1	√	√	√	√	√	√	√	√	√	√
2	√	√	√	√	√	-	√	√	√	√
3	√	√	√	-	√	√	√	√	√	√
4	√	√	√	√	√	-	√	√	√	√
Success	4	4	4	3	4	2	4	4	4	4
Completion Rates	100%	100%	100%	75%	100%	50%	100%	100%	100%	100%

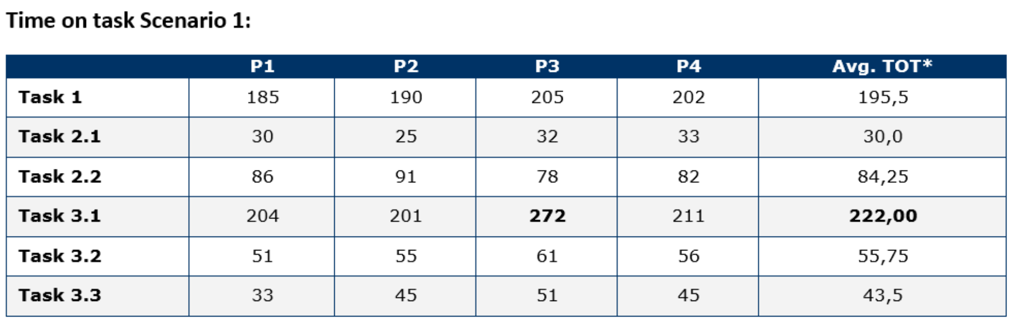
Figure 9.2: Scenario 2 Completion Rate

Time on task

Time on task for each participant was recorded. Some tasks were inherently more difficult to complete than others and is reflected by the average time on task. Time on task results for Scenario 1 is presented in Figure 9.3, and time on task results for Scenario 2 is presented in Figure 9.4.

Time on task - Scenario 1

- **Task 1(Start the application and set active team/repository)** showed a high average time on task. The main reason behind this was the authorization with GitHub which required users to login and authorize on the external GitHub.com page.
- **Task 2.1(View notifications)** showed a very similar time on task by the participants, the same can be seen on **Task 2.2(Find and archive notification)** although the time by each participant was over 80 seconds.
- **Task 3.1(Find reminder and perform daily reflection)** took the longest time to complete(average 222 seconds). However this was to be expected, as the daily reflection note requires participants to reflect on their work, and usually lasts for 2-5 minutes. This task also had the largest range in completion time, ranging from 204 seconds to 272 seconds. **Task 3.2(Find and share reflection note)** and **Task 3.3(Find mood graph)** participant time on task averaged 55 and 43 seconds.

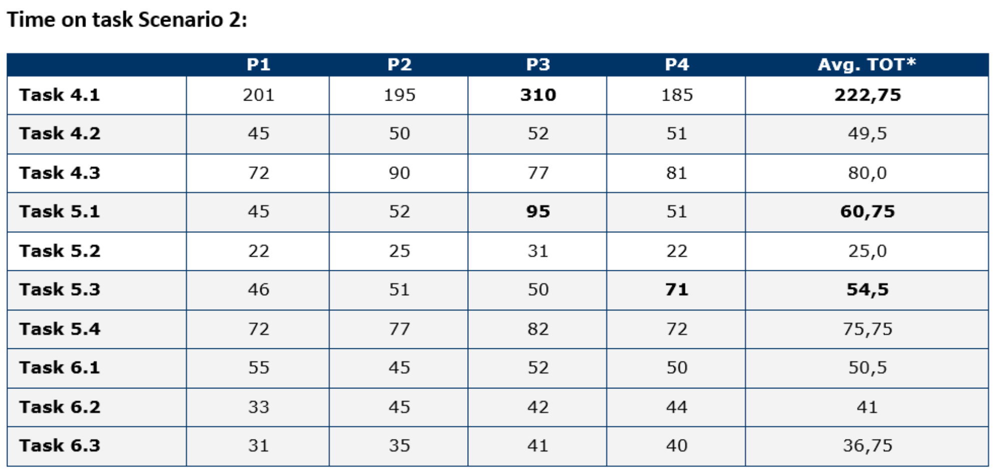

Time on task Scenario 1:

	P1	P2	P3	P4	Avg. TOT*
Task 1	185	190	205	202	195,5
Task 2.1	30	25	32	33	30,0
Task 2.2	86	91	78	82	84,25
Task 3.1	204	201	**272**	211	**222,00**
Task 3.2	51	55	61	56	55,75
Task 3.3	33	45	51	45	43,5

Figure 9.3: Time on task Scenario 1

Time on task - Scenario 2

- **Task 4.1(Create a team)** showed the longest completion time in Scenario 2. The main reason behind this was the amount of data that needed to be collected from GitHub and stored in the PeacefulBanana database. The task also showed the longest range in completion time, from 185 seconds to 310 seconds. The reason behind this large gap was mainly the difference in amount of data present in the GitHub repositories they chose to create a team for. **Task 4.2(Set active team) showed no major changes in completion time, and the same can be seen in Task 4.3(Identify team members)**
- **Task 5.1(Find all milestones)** showed that 3 out of 4 participants had very similar completion times(45, 52 and 51 seconds), but the average was increased by that the last participant had a completion time of 95 seconds. **Task 5.2(Identify overdue milestones)** showed very similar completion times, while **Task 5.3(Find repository issues)** had one participant at 71 seconds, while the rest used an average of 50 seconds. **Task 5.4(Find issue #17)** showed an average of 75 seconds, with no significant differences.
- **Task 6.1(Generate tag cloud)** averaged on 50 seconds, **Task 6.2(Identify most used individual and team tags)** averaged 41 seconds and **Task 6.3(Find commit impact)** averaged at 36 seconds, all with no significant difference in completion time.



Time on task Scenario 2:

	P1	P2	P3	P4	Avg. TOT*
Task 4.1	201	195	**310**	185	**222,75**
Task 4.2	45	50	52	51	49,5
Task 4.3	72	90	77	81	80,0
Task 5.1	45	52	**95**	51	**60,75**
Task 5.2	22	25	31	22	25,0
Task 5.3	46	51	50	**71**	**54,5**
Task 5.4	72	77	82	72	75,75
Task 6.1	55	45	52	50	50,5
Task 6.2	33	45	42	44	41
Task 6.3	31	35	41	40	36,75

Figure 9.4: Time on task Scenario 2

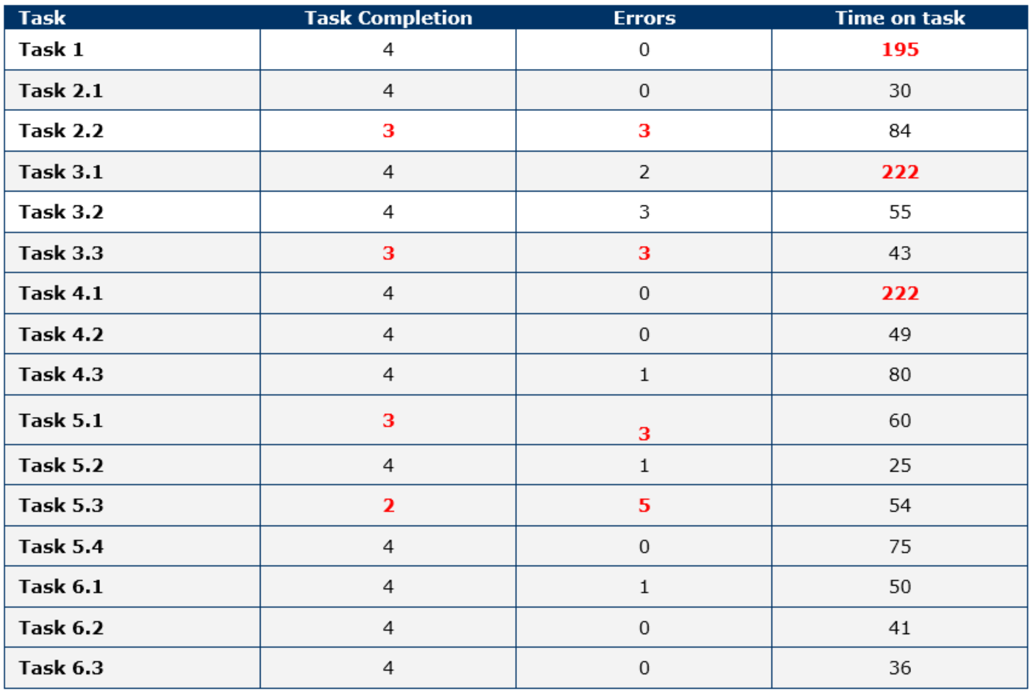
9.1.11 Summary of Data

The number of errors participants made while trying to complete the tasks were captured and recorded. Critical errors leads to participant failing in completing scenario, while non-critical errors is an error that does not prevent successful completion of the scenario. These errors along with task completions and time on task average for each task are represented in Figure 9.5. Low completion rate, occurrence of critical-errors and high time on task are highlighted in red.

Overall Metrics

After completing the usability session, participants were given a *System Usability Scale* form to answer[Brooke, 1996]. The results from the SUS-scale can be seen in Figure: 9.6.

All participants agreed(i.e., agree or strongly agree) that they would use the application frequently and that the application was easy to use. All participants(100%) also felt confident when using the application. The majority of the participants(75%) felt the functions in the application were well integrated, and that most people would learn to use the system very quickly. Half of the participants(50%) agreed that there were inconsistencies in the system, which were mainly related to a more clear distinction between milestone related issues and repository issues. None of the participants(0%) found the



Task	Task Completion	Errors	Time on task
Task 1	4	0	195
Task 2.1	4	0	30
Task 2.2	3	3	84
Task 3.1	4	2	222
Task 3.2	4	3	55
Task 3.3	3	3	43
Task 4.1	4	0	222
Task 4.2	4	0	49
Task 4.3	4	1	80
Task 5.1	3	3	60
Task 5.2	4	1	25
Task 5.3	2	5	54
Task 5.4	4	0	75
Task 6.1	4	1	50
Task 6.2	4	0	41
Task 6.3	4	0	36

Figure 9.5: Summary of Data

system unnecessarily complex or that it was cumbersome to use. Further none of the participants felt users would need support of a technical person to use the system(based on the intended user group) or that they needed to learn a lot before getting going with the system. The participants mentioned the quick start guide as a possible look-to document in case of trouble.

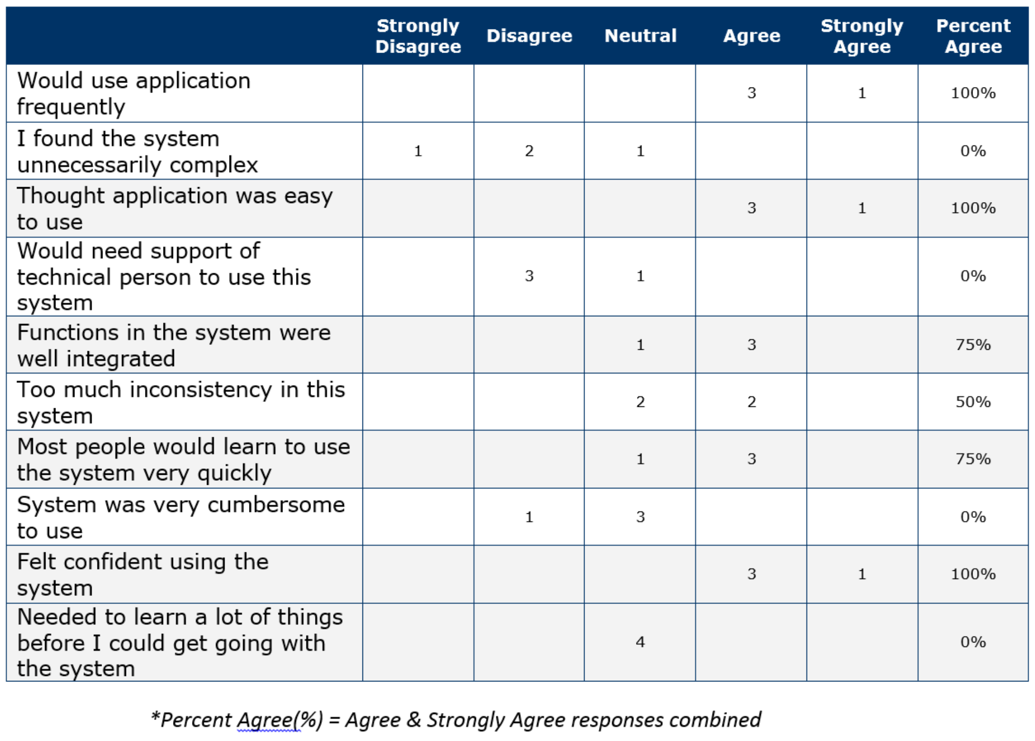
Reflection and sharing

Participants were also asked to answer the questions identified in Section 9.1.1

- Is the application easy to use, and can users achieve their goals in a timely manner?

Feedback here was that participants were satisfied with the ease of use as can be seen above, also time-on-task show that participants were mostly quite similar in the solving of tasks, and very few spikes.

- Identify the relationship users have with the aspect of reflection and sharing personal experiences.



	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Percent Agree
Would use application frequently				3	1	100%
I found the system unnecessarily complex	1	2	1			0%
Thought application was easy to use				3	1	100%
Would need support of technical person to use this system		3	1			0%
Functions in the system were well integrated			1	3		75%
Too much inconsistency in this system			2	2		50%
Most people would learn to use the system very quickly			1	3		75%
System was very cumbersome to use		1	3			0%
Felt confident using the system				3	1	100%
Needed to learn a lot of things before I could get going with the system			4			0%

**Percent Agree(%) = Agree & Strongly Agree responses combined*

Figure 9.6: Post task

On this context, participants answered that reflecting on their experiences is something they often do, but they don't collect them and thus forgets exactly what the experience was about. The application helped solve this problem by prompting and allowing users to reflect and then store it for later use.

When it comes to sharing, all participants were positive to this, although they strongly emphasized the need for an *unshare* functionality. The missing ability to unshare these notes after hand could make them reluctant to share them in the first place, since when first shared it was always shared. One participant mentioned the possibility of letting notes be editable and share/unshareable for a specific time period, f.ex 24 hours, where afterwards they would be locked for editing.

- Does the tool present data in a way that triggers reflection for the user?

Participants responded that the tag-cloud and questions in the daily reflection note triggered them to reflect on experiences. Actually reading the questions in their mind, helped them to reflect on the experiences, instead of just having an empty text-field could lead to random thoughts being collected and not actually triggering reflection. The commit impact graph was mentioned as less-helpful as it didn't really justify the amount of work done.

Participants also provided feedback for what they liked the most and least about the application, and recommendations for improving the application.

Most liked

The participants liked the design of the applicincation and that it was able to synchronize with their GitHub repositories automatically, without them having to worry about it. The personal tag-cloud and the ability to compare it directly with the team's tag cloud was also a joint feedback.

Least liked

Participants commented that the way notifications were given was not optimal, and the registration process was a bit tedious.

9.1.12 Recommendations

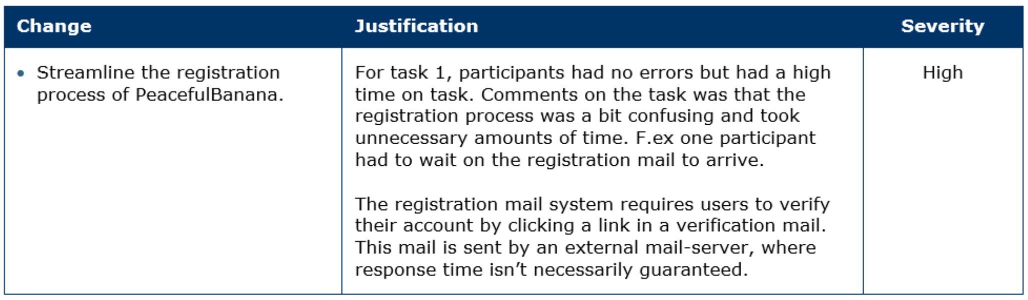
In addition to the feedback gathered from section 9.1.11, this section presents proposed changes and their justifications derived from the participant success rate, behaviors, and comments. The identified recommendations will improve the overall ease of use and address the areas where participants experienced problems or found the interface/information architecture unclear. Feedback on recommendations on improvement was primarily to streamline the registration process. Also the participants commented that issues connected to a specific milestone, should be more visibly separated from issues connected to the whole repository.

Implemented Changes

This section presents the proposed changes that were implemented into the PeacefulBanana application.

Task 1: Start application, login, link account with GitHub and set an active repository

This recommendation, it's justification and it's priority(Severity) is presented in Figure 9.7. The recommendation concerned a proposed streamlining of the registration process. This was applied to the application in the way that we changed to a larger and more stable mail-server provider, in order to reduce the significance of a slow response time from external providers. The registration process itself was also improved by reducing amount of clicks in the process.

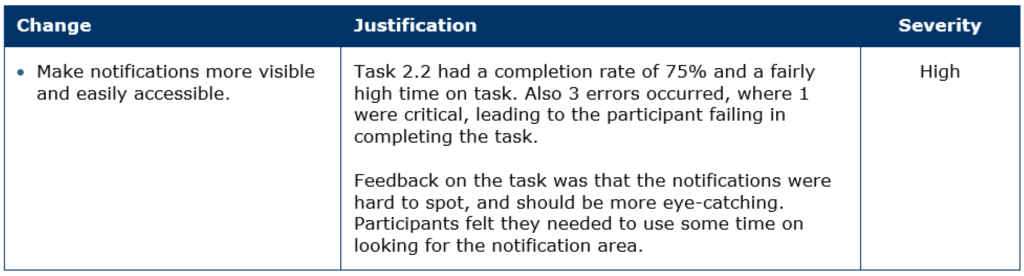


Change	Justification	Severity
• Streamline the registration process of PeacefulBanana.	For task 1, participants had no errors but had a high time on task. Comments on the task was that the registration process was a bit confusing and took unnecessary amounts of time. F.ex one participant had to wait on the registration mail to arrive. The registration mail system requires users to verify their account by clicking a link in a verification mail. This mail is sent by an external mail-server, where response time isn't necessarily guaranteed.	High

Figure 9.7: Task 1 changes and justification

Task 2.2: Find the *Congratulations* notification and archive it. Verify activation.

This recommendation, it's justification and it's priority(Severity) is presented in Figure 9.8. Implementation of the change resolved in adding a notification counter with a more vibrant color. The reason for implementing this change is that if users don't realize they have notifications, the message in the notification is lost and could lead to i.e. a user not remembering to do the daily notification, or that he has changed his team. This was the case in the usability test, which lead to one participant failing to complete the task.

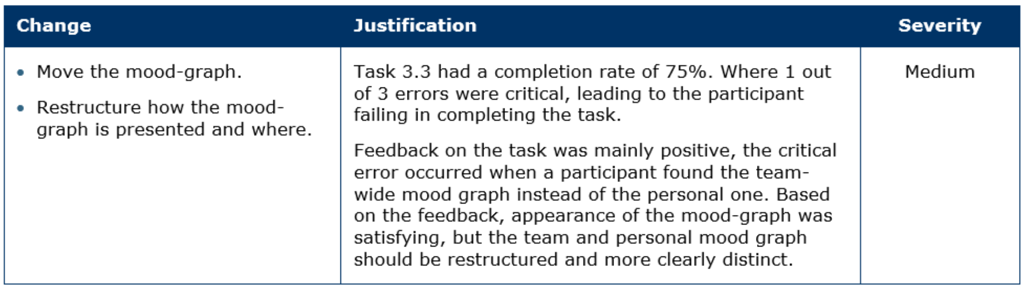


Change	Justification	Severity
• Make notifications more visible and easily accessible.	Task 2.2 had a completion rate of 75% and a fairly high time on task. Also 3 errors occurred, where 1 were critical, leading to the participant failing in completing the task. Feedback on the task was that the notifications were hard to spot, and should be more eye-catching. Participants felt they needed to use some time on looking for the notification area.	High

Figure 9.8: Task 2.2 changes and justification

Task 3.3: Find the mood-graph.

This recommendation, it's justification and it's priority(Severity) is presented in Figure 9.9. This recommendation was implemented by more clearly dividing the two different mood-graphs. This meant changing labels to implify what section they were connected to, and making the user more aware of what section they were in, and what tag-cloud they were looking at.

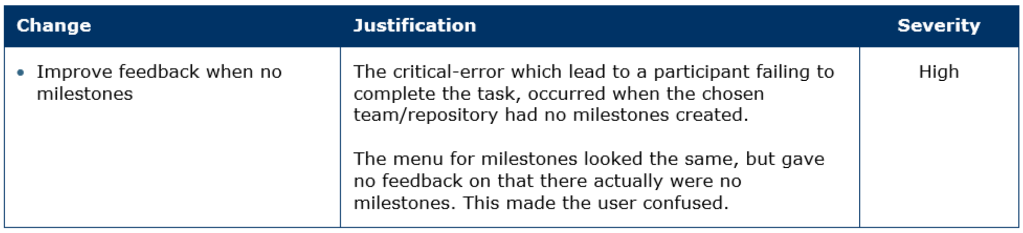


Change	Justification	Severity
• Move the mood-graph. • Restructure how the mood-graph is presented and where.	Task 3.3 had a completion rate of 75%. Where 1 out of 3 errors were critical, leading to the participant failing in completing the task. Feedback on the task was mainly positive, the critical error occurred when a participant found the team-wide mood graph instead of the personal one. Based on the feedback, appearance of the mood-graph was satisfying, but the team and personal mood graph should be restructured and more clearly distinct.	Medium

Figure 9.9: Task 3.3 changes and justification

Task 5.1: Find all milestones for the current active repository.

This recommendation, it's justification and it's priority(Severity) is presented in Figure 9.10. This improvement concerned missing feedback, when no milestones were made for a project. This lead to one participant failing to complete the task and therefore had a high priority. The improvement was implemented by adding a check to provide the user with appropriate feedback, if the chosen project had no milestones.

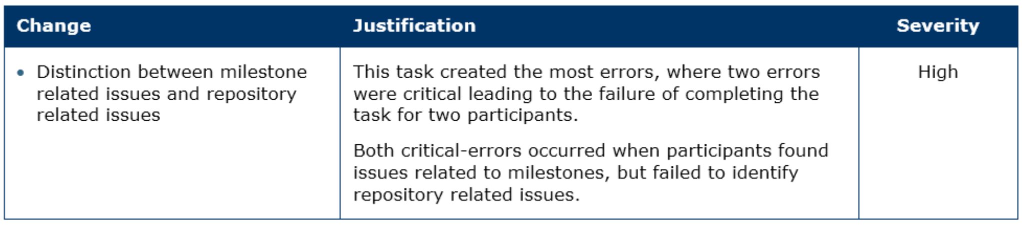


Change	Justification	Severity
• Improve feedback when no milestones	The critical-error which lead to a participant failing to complete the task, occurred when the chosen team/repository had no milestones created. The menu for milestones looked the same, but gave no feedback on that there actually were no milestones. This made the user confused.	High

Figure 9.10: Task 5.1 changes and justification

Task 5.3: Find repository related issues.

This recommendation, it's justification and it's priority(Severity) is presented in Figure 9.11. Task 5.3 generated the most errors, and also two critical errors leading to failure of completion. Therefore the improvement was given a high priority, and was implemented by clearly distincting milestone issues and the more general repository issues. We moved milestone issues to be more clearly visible under the *Milestone sub-menu*, and added the repository issues under the *Repository sub-menu.*



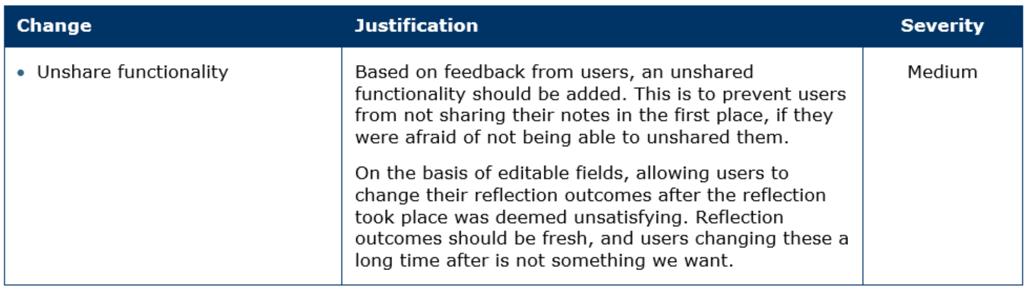
Change	Justification	Severity
• Distinction between milestone related issues and repository related issues	This task created the most errors, where two errors were critical leading to the failure of completing the task for two participants. Both critical-errors occurred when participants found issues related to milestones, but failed to identify repository related issues.	High

Figure 9.11: Task 5.3 changes and justification

Unimplemented Changes

This section presents the proposed changes which were not implemented to the PeacefulBanana application, and their justifications are presented in Figure 9.12 and Figure 9.13. The *unshare functionality change* was omitted, since allowing user's to change their reflection outcomes a long time after they occured, would not improve reflection for user's but instead obscure the outcome. The *commit impact change* was omitted, because of it's low priority. User's generally had a bad attitude towards what the commit impact captured, but reported they would rather have it there than not, as long as the team is aware that it might not reflect the correct project activity.

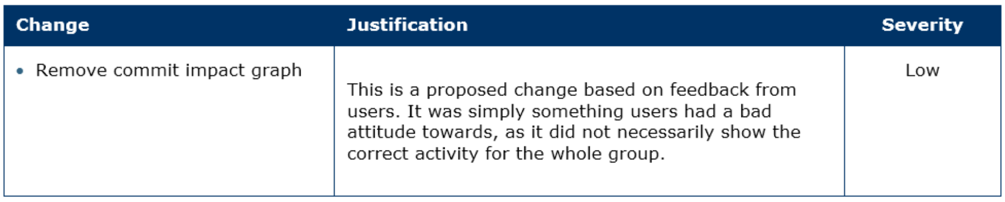
Add unshare functionality



Change	Justification	Severity
• Unshare functionality	Based on feedback from users, an unshared functionality should be added. This is to prevent users from not sharing their notes in the first place, if they were afraid of not being able to unshared them. On the basis of editable fields, allowing users to change their reflection outcomes after the reflection took place was deemed unsatisfying. Reflection outcomes should be fresh, and users changing these a long time after is not something we want.	Medium

Figure 9.12: Adding unshare functionality to reflection notes.

Remove commit impact



Change	Justification	Severity
• Remove commit impact graph	This is a proposed change based on feedback from users. It was simply something users had a bad attitude towards, as it did not necessarily show the correct activity for the whole group.	Low

Figure 9.13: Remove commit impact.

9.1.13 Conclusion

Most of the participants found the PeacefulBanana application to be well-organized, comprehensive, clean and uncluttered, very useful, and easy to use. Having an application to handle reflection in their daily work was desirable for all of the participants. Implementing the recommendations and further feedback from users will ensure a continued user-friendly application.

While some tasks had a high time-on-task, this was due to GitHub data collection, and is not something we have control over. It is a simple request to the GitHub API, which we need to receive before being able to continue. The application does not freeze during this wait, and allows users to continue using it, but the data collected is required for certain parts of the functionality.

9.2 Expert Review

9.2.1 Overview

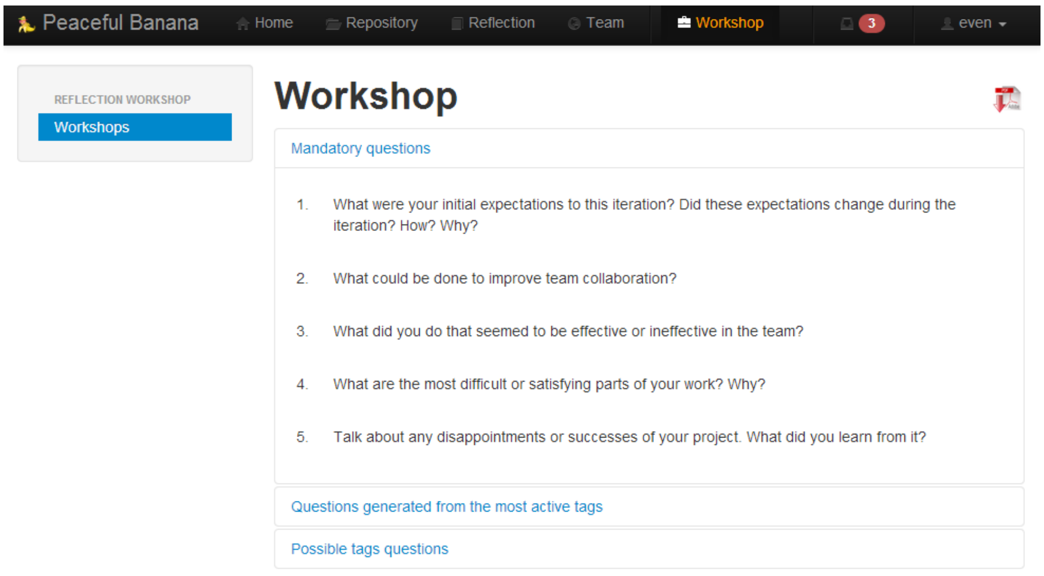
The expert evaluation was conducted with a senior scientist at Sintef Information and Communication Technology[4], which also has a position as adjunct associate professor at NTNU. The evaluator is an expert in the field of agile software development and knowledge management in software companies. The evaluator has published several case studies of agile teamwork in the software development industry. He also had knowledge of version-control systems, GitHub and the use of such tools in agile development teams. Apart from the expert review, the evaluator was never directly involved in our thesis.

The evaluation performed was a type of expert walk through as described in *Interaction Design - Beyond Human Computer Interaction*[Rogers et al., 2011]. The evaluation we performed differs in that we also evaluate how the application can support agile software development teams and reflection through revisiting experiences.

The evaluation started with us presenting the main features of the application and then continued with a walk through of the application in its production state. The walk through consisted of performing the typical tasks in our scenarios. After the walk through we had an open discussion around the most common challenges agile development teams meets. The evaluator also commented on possible shortcomings or limitations, and also any advantages the evaluator had identified in our application. We asked the evaluator to present his ideas on how our application could improve reflection in agile software development teams. We wanted an objective evaluation so we did not initially present any of our own thoughts regarding the application and its goals.

The evaluator suggested several possible limitations in the application, and also commented on how our application met some of the problems that often arise in development teams and how he could see our application potentially help limit these problems. We particularly went through the reflection workshop questions, to get feedback on the feasibility of these and triggering reflection in a retrospective session [See Figure 9.14].

[4]Sintef ICT: `http://www.sintef.no/home/Information-and-Communication-Technology-ICT/`



Workshop

Mandatory questions

1. What were your initial expectations to this iteration? Did these expectations change during the iteration? How? Why?
2. What could be done to improve team collaboration?
3. What did you do that seemed to be effective or ineffective in the team?
4. What are the most difficult or satisfying parts of your work? Why?
5. Talk about any disappointments or successes of your project. What did you learn from it?

Questions generated from the most active tags

Possible tags questions

Figure 9.14: Example of retrospective session questions to trigger reflection.

In addition to advantages and limitations of our application, the evaluator had some input towards related work he had seen, and how we could conduct the final evaluation in the best way. This included possible questions that might be relevant to ask the participants, how to get the best possible output from these and also some theory on how to analyze the results we got.

The feedback we got from the expert showed that the application had features that met many of the problems he had encountered through his studies of agile teams, and specifically how to trigger and enhance reflection. The evaluator also had some valuable feedback on possible shortcomings in the application, which can be typical challenges developers face in a day to day working environment.

9.2.2 Overall Feedback

The evaluation with the Sintef expert left us with the impression that the evaluator was satisfied with the general functionality of the application, in terms of agile teams and reflection in these teams. At the point of evaluation, the application had been deployed to production state, and so most of the functionality was in place. The evaluator stated that the choices we had made on the data collection and representation of these was satisfying, as it allowed and encouraged users to reflect on their experiences, while not being too intrusive on the daily work routine. Both in aspects of individual and collaborative reflection the application and its functionality was satisfactory.

As for integration of the tool, the evaluator saw a limitation in that it was generally hard to get new tools into the daily routine of developers[Rogers, 2010], and referred to the Technology Adoption Curve by Rogers[Figure 9.15. The way we encourage users to use *#hash tags* to tag important elements in the commit message, might take some time to work in. The evaluator expressed that he was happy with the design choices made and that we chose a web-application as a platform. This way the application is available for all individuals in the team, on a wide variety of devices. This availability is important in order to further lower the threshold of usage. Apart from general feedback and app-specific feedback we also got some feedback regarding the final evaluation. Specifically what questions to ask, comparing their previous retrospective routines with a retrospective using our application beforehand. Also feedback on how to properly analyze the results we would get was valuable. The evaluator had identified possible reasons for why a certain outcome could occur through his studies. The evaluator was also pleased with the notion of allowing teams to see what notes are shared , which allowed for identification of *sharing patterns* among the users. This is something that could help encourage a *"discussion about reflection in the retrospective sessions"*. Another point was that individual users tend to use the same tools in different ways, so specifying what and how the application could be used before users started using the tool was important.

9.2.3 App-specific feedback

Here we will present the challenges the evaluator presented in the aspect of teamwork and reflection in agile software development teams. The challenges identified in Table 9.1 was used to create a discussion around the Peaceful-Banana application and how it can solve these problems.

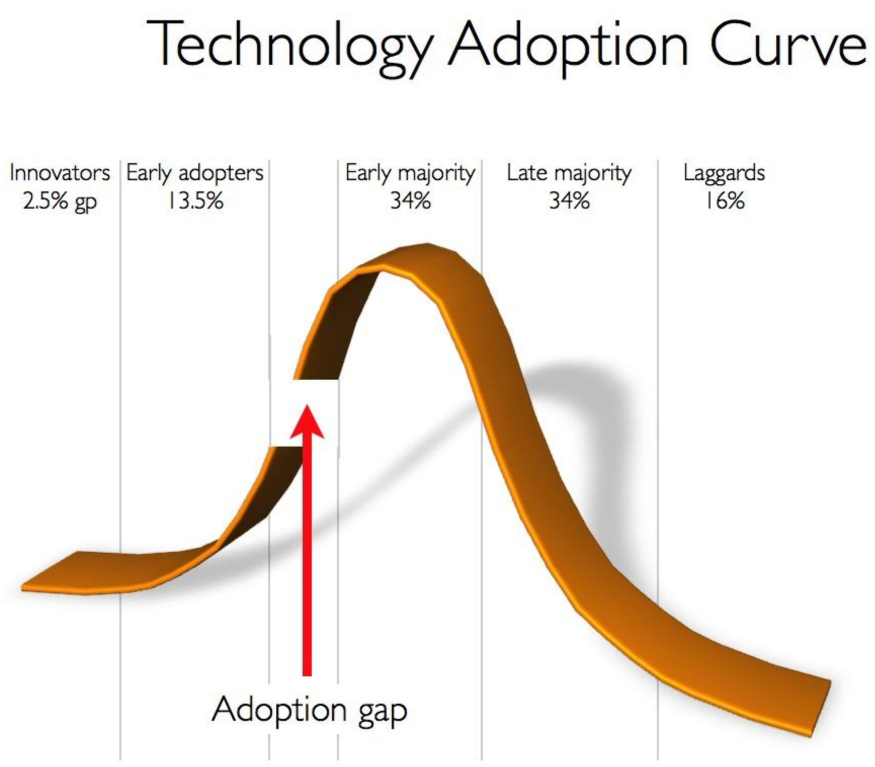


Figure 9.15: Roger's Innovation Adoption Curve [Rogers, 2010]

ID	Name	Description
1	Non-intrusive	The threshold of integrating new tools into the routines of software developers is hard. The evaluator specifically referred to the *Technology Adoption Curve* presented by Rogers[Rogers, 2010], which refers to the chasm between innovators or early adopters and the early majority. This curve can be seen in Figure 9.15.
2	Uniqueness	The application should meet a demand which hasn't already been met. Also the application should provide something that a normal retrospective does not.
3	Agile integration	How can the application be integrated into an agile environment, helping the team to be agile and not removing the agility from the team.
4	Dynamic Memories	Memories are dynamic and change over time, so there can be a lack of memorizing all important situations in a retrospective.
5	Priorities	Often agile teams develop what the developers are motivated for, and not what the customer prioritizes highest. These wrong-placed priorities can be hard to pick up.
6	Competence-overlap	Agile teams are most efficient and deliver the highest quality work when at least two people have the same competence, so that one can ask for help and code can be reviewed by a peer. When a developer is left alone on a piece of work, integrating these parts with the rest of the project can be an issue
7	Re-work:	Re-doing the same piece of work is also a challenge development teams can meet. When developers constantly revisits work that already has been accepted, to make unnecessary changes, the progress of the project is slowed down. Detection of this can allow for a discussion and allowing the team to progress.
8	Level of expertise:	Developers often have different levels of expertise, and different areas of expertise. Even though a developer have a high amount of impact on the code-lines committed to a project, this does not mean the others don't do important work

Table 9.1: Expert review feedback

- 1. Non-intrusive:

The evaluator was satisfied with the design choice we made. By keeping the amount of time users are **required** to put into the application to a minimum, the threshold of usage is kept as low as possible at the same time. As mentioned, the Roger's adoption curve state that integrating a new tool into a daily routine is hard already, so it's important the users don't feel the application is a necessity but a helpful tool. The evaluator specifically mentioned that the daily reflection note was a good choice, since it only takes roughly 5 minutes and serves a purpose for the users. If the users then should wish to dive further into the functionality, it is easy accessible.

- 2. Uniqueness:

The feedback here was that the evaluator had not encountered a similar tool during his research, and he felt the application met a need in the industry. As for the retrospective aspect, the feedback was that the application had features for identifying issues and situations that in a normal retrospective could be lost.

- 3. Agile integration:

The evaluator expressed a certain concern that too much data collection could lead to an overhead in the amount of data that needed to be analyzed during the retrospective. Emphasis here was that the application shouldn't come in the way of the team being agile.

- 4. Dynamic Memories:

The evaluator stated that by implementing the daily reflection note feature, we allowed for the most important experiences of the day being collected and stored for later use. This means that although not all data or experiences are collected, the application encourages users to reflect on fresh memories and can revisit these later during the retrospectives, hopefully omitting the risk of forgetting certain situations.

- 5. Priorities:

The evaluator stated that this is a common problem in development teams, and thus allowing for identification of such mis-priorities are important. The feedback was that the application features for seeing how far the team has come in the particular milestones, and going into the separate issues to see when it has been worked on and by who, partially had met this challenge. The evaluator still expressed that this could be even more emphasized, and be made more easily accessible.

- 6. Competence-overlap:

Through the different tag clouds featured in the application, a missing overlap in competence can be identified. The evaluator stated that comparing the team tag cloud with the personal tag cloud was a satisfactory way of identifying whether an individual has been working a lot alone. Further feedback was that some sort of comparing tag clouds between individuals would further help towards this challenge, but this could in turn lead to unwanted focus on individuals performance.

- 7. Re-work:

Also here the feedback was that inspection of issues allows for seeing when and how often a problem has been fixed and then re-opened again. Also by comparing tag clouds from different periods allow for identification of much re-work. Even though the functionality was there, the evaluator wanted to make these comparisons more easily accessible in the application.

- 8. Level of expertise:

The feedback here was: Since the application focuses on project commits and comparing the users activity based on the work committed to GitHub, the application could give the wrong impression of how much work individuals in the team have done. Because of this the evaluator emphasized that we kept this fresh in mind, when describing what the tool is, what it does, what it measures and most importantly what it doesn't measure. This is important so that no users feel their work is diminished in importance by using the application.

9.2.4 Suggested new features

During our discussion with the evaluator we identified some features that were missing which could be fitting to implement in the application:

- *What is new?* functionality:

Show parts of the source-code in the PeacefulBanana application, creating a sort of *What has happened since your last visit* functionality to the users and teams. The evaluator proposed that having such functionality might increase the user's motivation to use the application, and further trigger reflection for the daily reflection notes.

- Burn down-Chart:

The evaluator proposed including a burn down-chart based on the issues already existing on GitHub and in the PeacefulBanana application. A burn down chart is a graphical representation of work left to do versus time. The tasks or issues remaining(the backlog) is often on the vertical axis, with time along the horizontal axis.

A burn down chart is useful for estimating or predicting when all of the work or issues will be completed. In agile software development teams, it is a common tool to measure progress over time. The application supports some progress viewing in each milestone by presenting users with a progress bar, although the evaluator stated that a burn down-chart feature would be an addition teams would use and such increase the motivation for using the application as a whole. An example of such a burn down-chart can be seen in Figure 9.16. These features were noted as feature work, as implementing these could prove valuable for future users.

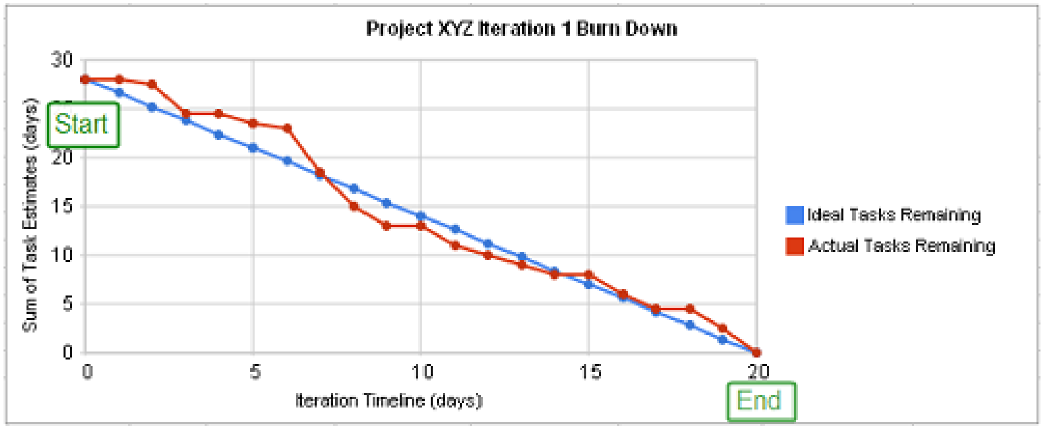


Figure 9.16: Example of a project burn down-chart [Wenzel, 2013]

9.3 Focus Group

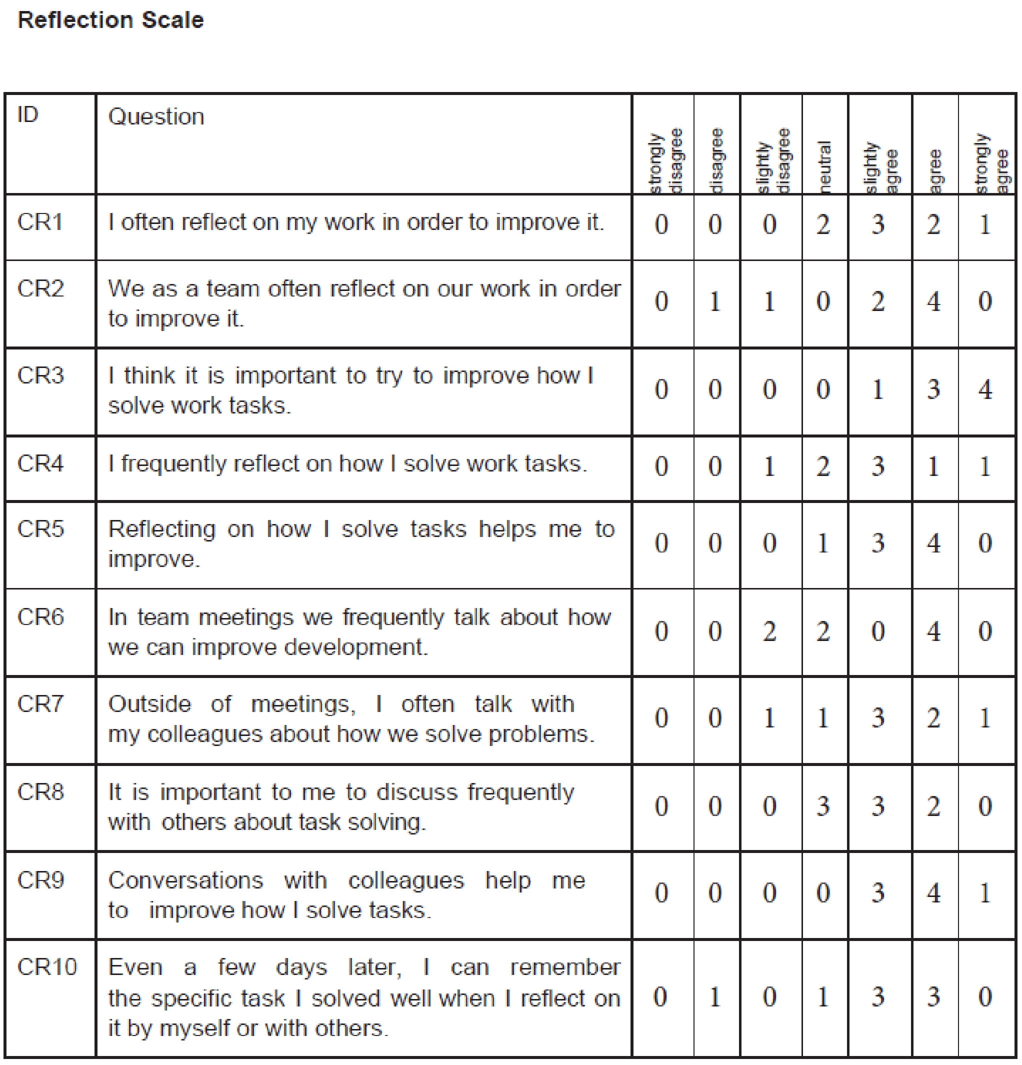
Focus groups is a form of qualitative research which can be compared to semi-structured group interviews[Rogers et al., 2011]. Participants in a group are asked about their opinions and views towards a product or concept based on their background and experiences[Krueger and Casey, 2008]. The group should consist of six to ten persons, where participants are not inhibited to present their honest opinions and experiences[Krueger and Casey, 2008]. The discussion is governed by a facilitator which have the task of keeping focus on the discussion in order to get answers to the questions that have been prepared beforehand[Krueger and Casey, 2008; Nielsen, 1997]. The facilitator is responsible for keeping the discussion open, uninhibited, non-judgmental and also making sure that all participants are allowed to present their views[Powell and Single, 1996].

A focus group usually lasts between 90 and 120 minutes, and is conducted in a neutral place. Limitations of a focus group include that you only get information on what potential users say, and not what they do or if it aligns with the reality[Nielsen, 1997]. Another limitation is that users often think they need something else than what they really need. Therefore it is important to demonstrate something concrete, where users understand fully what the application is, what it does and what it achieves. The theory above and the guide proposed by Krueger [2002] provided us with a basis on how to conduct a focus group interview. We gave the participants a smooth and snappy introduction to the agenda of the interview.

9.3.1 Focus group context

The application was evaluated with a focus group consisting of 8 NTNU students, working together in a group in the IT2901 bachelor - project. The group were using scrum as their agile project methodology, and have also used it previously in development projects. A group of 8 is fairly big, although larger focus groups are recommended in order to collect more commentaries and details from discussions[Morgan et al., 1998]. Participants in the focus group were familiar with the use of GitHub and were also using GitHub for their project at the time of evaluation. In their project they use retrospectives after each sprint, since they are using scrum. This provides the focus group with participants eager to improve their collaboration in agile teams and to improve reflection, both individually and in teams during retrospective sessions. The focus group was hosted at NTNU, in a private workshop lab with a circular table.

The group were given the reflection scale from the MIRROR evaluation toolbox before starting the evaluation. Their answers and relationship to reflection can be seen in table 9.17.



Reflection Scale

ID	Question	strongly disagree	disagree	slightly disagree	neutral	slightly agree	agree	strongly agree
CR1	I often reflect on my work in order to improve it.	0	0	0	2	3	2	1
CR2	We as a team often reflect on our work in order to improve it.	0	1	1	0	2	4	0
CR3	I think it is important to try to improve how I solve work tasks.	0	0	0	0	1	3	4
CR4	I frequently reflect on how I solve work tasks.	0	0	1	2	3	1	1
CR5	Reflecting on how I solve tasks helps me to improve.	0	0	0	1	3	4	0
CR6	In team meetings we frequently talk about how we can improve development.	0	0	2	2	0	4	0
CR7	Outside of meetings, I often talk with my colleagues about how we solve problems.	0	0	1	1	3	2	1
CR8	It is important to me to discuss frequently with others about task solving.	0	0	0	3	3	2	0
CR9	Conversations with colleagues help me to improve how I solve tasks.	0	0	0	0	3	4	1
CR10	Even a few days later, I can remember the specific task I solved well when I reflect on it by myself or with others.	0	1	0	1	3	3	0

Figure 9.17: Reflection scale results

The group generally agree with the importance of reflection overall in the reflection scale questionnaire, however they disagree a bit in regards to team-reflection. This is quite interesting, but during the interview we noticed a few very dominant figures in the team and this might be the reason why the answers vary a bit regarding team-reflection.

9.3.2 Ground Rules

In the introduction we identified a set of ground-rules that would make the interview go smoother and keep the participants distraction free. Most of these are based on the ground rules defined in Krueger [2002], and can be seen in the list below.

1. Mobile phones are to be turned off during the entire interview. If you are not able to do so and receive a call ,please do this as quietly as possible.
2. We are recording the session on tape, so only one person should speak at a time.
3. There is no right or wrong answer here, only opinions.
4. You do not need to agree with others, but you must listen respectfully as others share their views.
5. Talk to each other, not directly to the evaluator.

9.3.3 Data Collection and Analysis

The session was recorded, in order to analyze the results after hand and also to be able to participate actively during the walkthrough. We had prepared some application-specific questions to ask the participants during the session. These questions consisted of the most relevant app-specific questions from Section 9.1.3 in the MIRROR evaluation toolbox [Kristin Knipfer, 2012]. These questions were slightly modified to be used as part of the focus group and can be seen in Table: 9.2

1	Can PeacefulBanana help you to collect information relevant to reconstructing experiences from work?
2	Can the application help you to reflect on experiences from work?
3	Can the application help you to capture reflection outcomes?
4	Does the application help you by making daily reflection notes available for later use?
5	Does the application remind you to reflect on experiences?
6	Does the application help you to find relevant experiences from others in your team?
7	Does the application help you to remember previous experiences and reflections?
8	Does the application help you to store information regarding work experiences?
9	Can the application help you to decide if and when to reflect?
10	Do the application help you by supporting the sharing of experiences?
11	Does the application guide you on how to share experiences with others?
12	Can the application improve your collaboration?
13	Does the application provide relevant content for reflection?
14	Did the application guide you through the reflection process?

Table 9.2:

After a brief presentation of the application itself and our goal with this thesis, the walkthrough started. The goal of the evaluation was primarily to see how an agile development team could integrate our tool into their daily routine. The application was evaluated with test-data, so participants could easily see how it would function in a daily work environment. After showcasing the different features, the researchers facilitated the focus group discussion on how the application could promote reflection for the team, but and also potential shortcomings or challenges to integrating it in their daily routine. The questions asked during the focus group were largely open-ended, which allowed participants freedom to express their views on the application[Yin, 2008]. The focus group was conducted in a responsive manner, allowing us to follow up on issues uncovered mid-session and adjust the content of the focus group based on this[Rubin and Rubin, 2011; Wengraf, 2001].

While one researcher facilitated the session, another observed and took notes, with timestamps of important parts. We swapped roles during the session, in order to prevent any variance in the notes and questioning. Any ambiguity was clarified with the participant before moving on. In order to aid analysis, the focus group was recorded and transcribed, before being analyzed. The group was also filmed so that their body-language could give us an indication about their involvement in the discussion and it was pointed out by the researchers that the interview would be anonymized to ensure that the statements could not be linked back the them. Quotes from participants are denoted with *FGX*, which stands for *Focus Group user X*. This was done in order to keep participants anonymous.

9.3.4 Why they did not use it?

The group was intended to participate in a case-study evaluation of the system over a period of time, but expressed that the stress level they where under during the test period and the amount of work they had remaining, made them focus on that rather than testing the tool for us. They also expressed that since the tool was made available for them so late in the process and the fact that they already had a routine worked in, they often forgot to include the tool in their routine on a daily basis. Additionally the group only got together and worked a couple of days a week, was a bit of a surprise to us, as we where expecting them to work almost full-time on the project.

9.3.5 If they would have used it

When discussing *tags*, the group stated that they felt this might become *'off topic'* and that team-members could use tags not relevant to the work at hand. This could in turn corrupt the tag clouds since *'off topic'* tags would gain magnitude if people used the same tags for whatever they commited. One of the participants expressed that he would have liked the possibility to tag an entire commit with a theme like [GUI] or [BACKBONE], which were something the other participants felt would be a great addition to the existing possibilities regarding tags.

> *Adding categories in brackets or something like that would put the hash tags in perspective too what people are working on and see how much time of their day goes to what part of the project. (FG1)*

This would allow tags to be categorized into different parts of the project, which we feel could add another dimension to the tool and help the team see more relevance in the tags used. However we feel that the themes or categories should be decided at project-/iteration-start, or at least agree on some rules for what tags can be used on what piece of work. The findings discussed implies that the group answered yes to question 2, 3 and 8 in table 9.2 with the addendum described above as a possible feature improvement.

The group also had a few pointers regarding the tag cloud. The group felt tag-clouds was a good representation, the relationship between the team tag-cloud and the individual tag-cloud was not easily identified. They proposed that tags needed to be placed at the same place and be in the same color in the tag-cloud, in order to allow for easier comparison. In the demonstration, the tags were neither placed in the same place or in the same color.

> *It is hard to locate the same tags in both of the tag clouds when they are placed in different regions of the cloud and in different colors. (FG2)*

It was also debated whether or not it would make any difference to weight tags based on the amount of work behind the commit. We all agreed that this could be a feature that could be implemented later. Some of the group members pointed out that even only one line changed could be just as important as an entire class being implemented.

> *One line to fix a bug, can be as important or even more important than a entire new class. (FG2)*

The group mentioned that while doing their retrospectives, they struggled to remember what they had done the previous weeks. When discussing daily reflection notes, they stated that this would give them an new dimension to ordinary retrospectives with the possibility to generate questions based on the work that had been done each day. This way, the most important work would be captured each day and easily remembered. The group also had some comments on how to possibly improve the feature: By adding a text field where the user could explain what he had done that day, might enhance the level of reflection. This was something they felt was missing from project-management tools like Trello[5] which they were already using. When we introduced the group to the feature *workshop preparation* we were overwhelmed by the positive response, but they also had some input on how it could be improved. The group stated that we should include the team-wide tag-cloud for the selected period. These features mentioned gave the group the possibility to reconstruct experiences recorded earlier and thus answering questions 1, 7 and 13 in table 9.2.

[5] `http://www.trello.com`

As the group got closer and closer to their final evaluation they experienced that their retrospectives got shorter and shorter, it also lacked structure. This corresponds with the findings described by [Kasi et al., 2008] in Section 1. When discussing the workshop-feature some of the group members commented that it could be great for a project manager that does not spend that much time with the code hands-on and in general a great tool for retrospectives, but since they had not tested the tool in they way they conduct work, there was no way for them to verify it. This suggests that the group are positive to question 14 in table 9.2.

> *I would imagine that this would be very useful for a project manager that does spend all day hands-on with coding. (FG1)*

9.3.6 Comments

The group had some comments and suggestions on how to improve the prototype. As for the daily reflection note feature, a suggestion was that instead of having the note non-editable and allow for only one note each day, users could edit the note whenever they wanted during the day and at the end of it, the note will be locked for user editing. This suggestion would give the user the option of adding to a note all day long. The downside of the change would be that the freshness of the reflection would be mitigated. When the reflection on that days experiences have taken place, allowing users to go back and change their input would take some of the value out of the reflection. The group suggested that the daily reflection note should also include a team tag-cloud, in addition to the commit impact cake-diagram and the individual tag-cloud.

The group also mentioned that the current notifications were not catching their attention, and suggested they be should be more dominating on the site, making the notifications impossible to miss. As they stated:

> *As notifications are there to remind users to do their daily reflection, missing the notification might lead to missing the daily reflection also.*

And:

> *The current notification is not invasive enough, it should be impossible to ignore. (FG3)*

This indicates that the group slightly disagreed to question 5 in Table 9.2, as they wanted the reminder to be more dominant and impossible to avoid.

It was also suggested that the group could have a meeting where they planned themes and tags to be used for the next iteration, in order to make the tool more *scrum specific*, as they put it, in terms of project management. Making the tool less invasive in the Scrum process is something we feel might ensure that users visit the tool more often, as it's not interfering as much with their daily routine.

One group member said it would be interesting to see if there is a connection between what the users are working on and their mood. I.e. what are users working on when they are in a bad mood or if a user is always unhappy when working with the graphical design, he should probably not work more on that part of the project.

> *Seems like a natural and good tool. (FG2)*

9.4 Discussion

In this section we will discuss the results and experiences gathered during the usability test, expert review and focus group evaluation of the PeacefulBanana application. We will discuss this in regard of our research questions PeacefulBanana were intended to answer. We will answer all the sub research questions first, since these are more specific by trying to answer parts of the main research question. Finally we will summarize the discussion by answering the main research question, since this is the most general in terms of reflection and learning from experiences.

9.4.1 Sub RQ1

How to scaffold collection of data in order to promote reflection?

The expert evaluator expressed a concern regarding the amount of data that needed to be processed and that it should not stop the team from being agile, but also pointed out that the feature itself could help during a retrospective session. He also pointed out that the daily reflection note provided the users with a quick and easy way to reflect on a daily basis and that this would work for both users in teams and individuals. However individuals would use the functionality differently than team-users. Team users would also have an effect by looking at the me-vs-team tag-cloud, and he pointed out that this could trigger reflection as well when comparing your own work with the rest of the team.

The focus group pointed out that they might wanted more data showed or at least the possibility to view data in their original form[6] in additions to data we provided for them. They expressed that the use of commit messages to generate questions for retrospectives would give them a another dimension and more structure.

9.4.2 Sub RQ2

How to increase the tendency to reflect on experiences, both individually and as a team?

Regarding the individually reflection the focus group expressed that a reminder in a tool like PeacefulBanana is a good way to trigger reflection, however they felt like the notification was a bit mild and should be more invasive to more force them to reflect over the last the 24 hours of work. Some group members expressed that the fact that the notifications needs to be read and that you can have multiple of them defeats the purpose of the notification, they are not interested in a historic overview of reflection reminders.

They felt that the tool did not provide them with any reminder in order to improve their reflection rate as a team, however some of them stated that the link only visible to team owner and manager could have some effect in order for them to plan a retrospective session.

9.4.3 Sub RQ3

How to bring together contributions from multiple users, and sharing these in a collaborative environment?

[6]Entire commit messages and issues titles in the tag cloud f.ex.

The focus group liked that the tool had the possibility to share notes with other team members and that these notes can be inspected. The expert evaluator stated that this was vital for the users to reconstruct work experiences in order for them to reflect over past experiences. The tag-cloud could also be used to identify how much you work on different issues.

The expert reviewer stated that the application introduced some new aspects that changed the normal routine for a retrospective, by generating suggestive questions based on the work done. He then expressed that it would have liked the application to be tested with a real agile software development team over a period of time and then conduct a retrospective session with that team. It would then be interesting to see if the laggers described in Figure 9.15(Section 9.2.2) would have as much to contribute with as the early adopters.

9.4.4 Main RQ

How to promote experienced-based reflection based on project artifacts collected from version-control systems?

Every step of the PeacefulBanana work-process have been evaluated by the different evaluators as described in the sections above, when discussing the different sub research questions. The first step is related to collecting data for reflection, which the application gathers from GitHub. The data is then analyzed and tags added from each member are displayed in their daily individual tag-cloud as well as the team-wide tag-cloud. The tag-clouds are here to remind the user of what they have been working on that day. During both evaluations it was expressed that collecting data from GitHub in order to identify trending issues within the team, was something the users could benefit greatly from.

The next step in the process is reflection. This step uses the data collected in order for users to revisit experiences and to trigger reflection on these. Allowing the user to compare his individual tag-cloud with the team-wide tag-cloud may further help trigger this reflection. This was further agreed upon by the results from the evaluations, even though the focus group participants had some issues with the tags in the tag-cloud was not distinct enough. The users are prompted to fill out a daily reflection note and this process is more scaffolded, providing a frame for users with a set of questions related to their daily activities.

The third step consists of sharing your experiences and reflection outcomes collected in the daily reflection notes with team members. This sharing of notes enables the whole team to inspect all the individual reflection outcomes and compare with their own. These issues can be further discussed collaboratively in the retrospective sessions.

The fourth and last step consists of taking the outcomes of your reflection and putting it to use in work related tasks in order to improve. This step mainly focuses on the *Workshop* functionality, providing the team with a frame which guides their retrospective. The frame consists of questions related to reflection in general and questions generated from their most active issues and tags. This frame ensures that the issues identified as most important during the iteration by the team-users are discussed and can be improved.

The evaluators expressed both concerns and appreciation for the use of these data. The fact that the expert reviewer would have liked to see a long-term case-study with the application is a great sign that the he finds the tool useful in the aspect of reflection in agile development teams.

Chapter 10

Conclusion

10.1 Summary

In this thesis we investigated how collection and scaffolding of project artifacts from the VCS GitHub can act as reflection triggers and possibly enhance reflection in software development teams. We have developed a platform-independent prototype in order to see how to visualize and present data collected from the artifacts.

During the last decade there has been a huge focus on improving development methodology and the now renowned industry-standard is agile development methodology. Agile software development projects generate a lot of data per developer in the version control system and other tools relevant for project management, each of the commit to the version control system contains tiny notes relevant to the work they have done. Our investigation shows that there is little to no other projects done based on this data and this was backed up during our expert evaluation, the scientist pointed out that during all his time around agile development projects he had never see any tools like the one developed by us. Even though that there has been a great deal of research regarding reflection in development teams. When we designed the prototype we set up a set of requirements which where prioritized based on difficulty and time to completion.

The implementation was a result of the theoretical background and related work, evaluation and discussion with an expert in the area of agile development. Then the resulting prototype was evaluated with a focus group consisting of a agile development team.

We then analyzed the results together with the theoretical background and notes from the expert review, where we discovered that the tools features was found useful. However we also discovered that some of the key features could be improved with the notes gathered from both the expert review and the focus group interview.

10.2 Discussion on our own work

The primary work done for this thesis was the development of the PeacefulBanana application. Developing PeacefulBanana challenged us to learn about developing applications for the web, and a large variety of different devices and platforms. We also got to explore web frameworks for building web-applications, specifically the Grails framework and the domain of responsive web design. In addition to working with new and upcoming technology like HTML5 and Twitter Bootstrap, we learned how to setup our own Ubuntu server, with Apache, Apache TomCat and SQL, all of which is highly relevant in the daily work of developers.

Working together on this thesis and application has been both fun and exciting, and has also given us valuable experience and knowledge we will benefit from in our future work as software developers.

During development, several changes and plugins were made available for the Grails framework, some of which could have had a positive impact on the application and how it works. Because of time limitations, it was not plausible to implement these so late in the process, but it is a pity the application could not benefit from some of these additions.

We first created our scenarios with evaluating the tool on software development teams in the IT2901 course at NTNU. Unfortunately the groups were very busy, and was not able to use the tool enough for us to conclude on the data gathered. Therefore we conducted a focus group with such a development team, where we gained valuable information on why they didn't use the tool as often, and also how they felt about the tool *if they had used it.* As it turned out, our two scenarios functioned well in this setting, although they were first intended for evaluation over a longer period of time. The main reason for the group not using the tool, was simply that they were very busy and the tool was available only at a late stage of their project. Feedback from the focus group was that if introduced with the tool from the beginning it would have been easier to remember to use it, as it would be part of their routine. Having the teams evaluate the application in a real-work setting over a longer time-period may have given us more results.

Although usability was not our top-priority when developing the application, the usability test with real life users showed that usability has a large impact. Although we gained a lot of valuable feedback from our usability test, we think that an even earlier usability test could have provided us more insight on how to make the application more attractive and usable for real-life users.

When we look back at the work we have done in this thesis, we are both happy and satisfied with what we achieved. We got to work with new technology and create something we never had worked with before. We got to dive into the inner workings of the most popular version-control system *GitHub*, and take part of the whole development process from initial ideas to mock-ups, development, testing, deployment and evaluation of the final product. All in all the application developed for this thesis is something we look back on with pride and satisfaction.

10.3 Future Work

As the interview with the focus group discovered there were several minor improvements both in how certain functions are implemented, but also in how the feature itself is designed visually.

Technically we would like the tool to have a tighter integration with the version-control-system so that the synchronization itself will go automatically.

As the focus group helped us discover that the daily summary should hold more information, cause you can not really derive what you have done the last 24 hours, and the feature it self could be improved so that the user can edit the note through out the day. However we feel that this could interfere with the reflection level you would achieve with the solution we have created since noting a lot as things occur is not reflecting. It is clear that the daily summary needs to be improved, the fact that it does not provide the user with what they feel is enough data. We feel that an improvement of the current solution and the solution described above. By letting the user note what they have done continuously and answer the reflection specific questions only at the end of the day, the user could note all day long and reconstruct their day more easily and still reflect on a daily basis.

Regarding tag clouds it was discovered that it was hard to find the same tags in the team vs my tag cloud, because the tags is placed randomly and with a random color in both clouds. So we would place tags in the same region and with the same color so it would be easier to see the relationship between the tag clouds.

In the long term it would be a great addition to record the reflection outcomes from the reflection workshops so that this would be available for later use.

Bibliography

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2002). Agile software development methods: Review and analysis.

Aeronautics, N. and NASA, S. A. (2013). nasa.gov homepage. `http://www.nasa.gov/`.

Ainsworth, S. (1999). The functions of multiple representations. *Computers & Education*, 33(2-3):131–152.

Ambler, S. (2013). Scrum sprint poster.

Ap, W. W. and Frey, R. (2013). Groovy and grails application development. `http://en.wikipedia.org/wiki/File:MVC-Process.png`.

Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. The XP Series. Addison-Wesley.

Beck, K. and Andres, C. (2004). *Extreme programming explained: embrace change*. Addison-Wesley Professional.

Beck, K., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Principles behind the Agile Manifesto.

Beck, K. e. a. (2013). Agile manifesto. `http://agilemanifesto.org/`.

Bjarnason, E. and Regnell, B. (2012). Evidence-Based Timelines for Agile Project Retrospectives – A Method Proposal. pages 177–184.

Boud, D., Keogh, R., and Walker, D. (1985). *Reflection: Turning Experience into Learning*. Routledge.

Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189:194.

Cellini, S. R. and Kee, J. E. (2010). Cost-effectiveness and cost-benefit analysis. *Handbook of practical program evaluation.*

Chaiklin, S. and Lave, J. (1993). *Understanding practice: Perspectives on activity and context.*, volume 0 of *Learning in doing.* Cambridge University Press.

Cockburn, A. (2006). *Agile Software Development: The Cooperative Game*, volume 113. Addison-Wesley Professional.

Derby, E. and Larsen, D. (2006). *Agile Retrospectives: Making Good Teams Great*, volume 1. Pragmatic Bookshelf.

Drury, M., Conboy, K., and Power, K. (2011). Decision Making in Agile Development: A Focus Group Study of Decisions and Obstacles.

Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.*, 50(9-10):833–859.

Esearch, S. Y. R., Hevner, B. A. R., March, S. T., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. 28(1):75–105.

Fleck, R. and Fitzpatrick, G. (2010). Reflecting on Reflection : Framing a Design Landscape.

Git (2013). Git – everything is local. distributed version control system. `http://www.git-scm.com`.

GitHub (2013a). Github - web-based hosting service for projects that use the git revision control system. `http://www.github.com`.

GitHub (2013b). Github 3 million users and 5 million repositories. `https://github.com/blog/1382-three-million-users`.

GitHub-API-v3 (2013). Github api version 3. `http://developer.github.com/v3`.

GitHub.com-TwitterBootstrap (2013). Github issue tracker - twitter bootstrap repository.

Gulliksen Stray, V., Moe, N., and Dingsøyr, T. (2011). Challenges to teamwork: A multiple case study of two agile teams. *People and computers*, 77:146–161.

Harrison, W. (2006). Eating your own dog food. *Software, IEEE*, 23(3):5–7.

Hearst, M. a. and Rosner, D. (2008). Tag Clouds: Data Analysis Tool or Social Signaller? *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pages 160–160.

Høyrup, S. (2004). Reflection as a core process in organisational learning. *Journal of Workplace Learning*, 16(8):442–454.

JGit (2013). Github api version 3. `http://www.eclipse.org/jgit/`.

Kasi, V., Keil, M., Mathiassen, L., and Pedersen, K. (2008). The post mortem paradox: a delphi study of it specialist perceptions. *European journal of information systems*, 17(1):62–78.

Ken Schwaber, J. S. (2011). Official scrum rulebook.

Kirkman, B. L. and Rosen, B. (1999). Beyond self-management: Antecedents and consequences of team empowerment. *academy of Management Journal*, 42(1):58–74.

Kolb D.A, R. (1975). Towards an applied theory of experiential learning. pages 33–58.

Korthagen, F. and Vasalos, A. (2005). Levels in reflection: core reflection as a means to enhance professional growth. *Teachers and Teaching Theory and Practice*, 11(1):47–71.

Kristin Knipfer, Daniel Wessel, K. D. (2012). Specification of evaluation methodology and research tooling.

Kristin Knipfer, Daniel Wessel, K. D. (2013). Scrum burndown-chart.

Krogstie, B. and Divitini, M. (2010). Supporting reflection in software development with everyday working tools. COOP:141–162.

Krogstie, B., Prilla, M., Wessel, D., Knipfer, K., and Pammer, V. (2012). Advanced learning technologies (icalt), 2012 ieee 12th international conference on. pages 151–153.

Krogstie, B. R. and Divitini, M. (2009). Shared Timeline and Individual Experience: Supporting Retrospective Reflection in Student Software Engineering Teams. *2009 22nd Conference on Software Engineering Education and Training*, pages 85–92.

Krogstie, B. R. and Prilla, M. (2011). Tool support for reflection in the workplace in the context of reflective learning cycles Background : Computer supported reflective learning. pages 57–71.

Krueger, R. A. (2002). Designing and conducting focus group interviews.

Krueger, R. A. and Casey, M. A. (2008). *Focus groups: A practical guide for applied research*. SAGE Publications, Incorporated.

Kwiecien, M. (2013). What is a retrospective?

Li, I., Dey, A. K., and Forlizzi, J. (2011). Understanding my data, myself: supporting self-reflection with ubicomp technologies. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 405–414. ACM.

Lin, X., Hmelo, C., Kinzer, C. K., and Secules, T. J. (1999). Designing technology to support reflection. *Educational Technology Research and Development*, 47(3):43–62.

Maham, M. (2008). Planning and Facilitating Release Retrospectives.

Marcotte, E. (2013a). Fluid grids. `http://alistapart.com/article/fluidgrids`.

Marcotte, E. (2013b). Fluid images. `http://alistapart.com/article/fluid-images`.

Marcotte, E. (2013c). Responsive web design. `http://alistapart.com/article/responsive-web-design`.

Markus, M. L., Majchrzak, A., and Gasser, L. (2002). A design theory for systems that support emergent knowledge processes. *MIS Q.*, 26(3):179–212.

MIRROR (2013). Mirror csrl model v1.2.1. `http://research.idi.ntnu.no/mirror/csrl_v1_2_1/CSRL_v1_2_1_Clickable_NTNU/start.html`.

Morgan, D. L., Krueger, R. A., and King, J. A. (1998). *Planning focus groups*. SAGE Publications, Incorporated.

Nielsen, J. (1997). The use and misuse of focus groups. *Software, IEEE*, 14(1):94–95.

Nielsen, J. and Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 206–213. ACM.

Nimesh, R. (2013). How fluid grids work in responsive web design. `http://www.1stwebdesigner.com/tutorials/fluid-grids-in-responsive-design/`.

Otto, M. (2013). Building twitter bootstrap. `http://alistapart.com/article/building-twitter-bootstrap`.

People10.com (2013). Groovy and grails application development. `http://people10.com/blog/groovy-and-grails-application-development-2/`.

Powell, R. A. and Single, H. M. (1996). Focus groups. *International journal for quality in health care*, 8(5):499–504.

Reenskaug, T. and Coplien, J. O. (2009). The dci architecture: A new vision of object-oriented programming. *An article starting a new blog:(14pp) http://www. artima. com/articles/dci_vision. html.*

Robin, X. (2011). Client-server model. `hhttps://en.wikipedia.org/wiki/File:Client-server-model.svg`.

Rogers, E. M. (2010). *Diffusion of innovations*. Free press.

Rogers, Y., Sharp, H., and Preece, J. (2011). *Interaction design: beyond human-computer interaction*. Wiley.

Rubin, H. J. and Rubin, I. S. (2011). *Qualitative interviewing: The art of hearing data*. SAGE Publications, Incorporated.

Scardamalia, M., Bereiter, C., McLean, R. S., Swallow, J., and Woodruff, E. (1989). Computer-supported intentional learning environments. *Journal of educational computing research*, 5(1):51–68.

Schindler, M. and Eppler, M. J. (2003). Harvesting project knowledge: a review of project learning methods and success factors. *International Journal of Project Management*, 21(3):219–228.

Schön, D. A. (1983). *The Reflective Practitioner: How Professionals Think in Action*, volume 1? Basic Books.

Schwaber, K. and Beedle, M. (2002). *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River.

Takeuchi, H. and Nonaka, I. (1986). The new new product development game. *Harvard business review*, 64(1):137–146.

Talby, D., Hazzan, O., Dubinsky, Y., Keren, A., and Force, A. (2006). Reflections on Reflection in Agile Software Development.

Tata, J. and Prasad, S. (2004). Team self-management, organizational structure, and judgments of team effectiveness. *Journal of Managerial Issues*, pages 248–265.

Twitter (2013). Twitter bootstrap. `http://twitter.github.com/bootstrap/index.html`.

US.gov (2013). Usability pilot test. `http://www.usability.gov/methods/test_refine/learnusa/preparation.html`.

Vinaganda.com (2013). What is responsive design. `http://vinaganda.com/what-is-responsive-design/`.

Wengraf, T. (2001). *Qualitative research interviewing: Biographic narrative and semi-structured methods.* SAGE Publications Limited.

Wenzel, J. (2013). Agile burn down chart.

Wieringa, R. (2009). Design science as nested problem solving. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, DESRIST '09, pages 8:1–8:12, New York, NY, USA. ACM.

Woerkom, M. V. and Croon, M. (2008). Operationalising critically reflective work behaviour. *Personnel Review*, 37(3):317–331.

Wright, P. and Monk, A. F. (1989). Evaluation for design. *People and computers*, 5:345–358.

Yin, R. K. (2008). *Case study research: Design and methods*, volume 5. SAGE Publications, Incorporated.

Appendix A

PeacefulBanana Quick Start

Figure A.1: PeacefulBanana Reflection tool

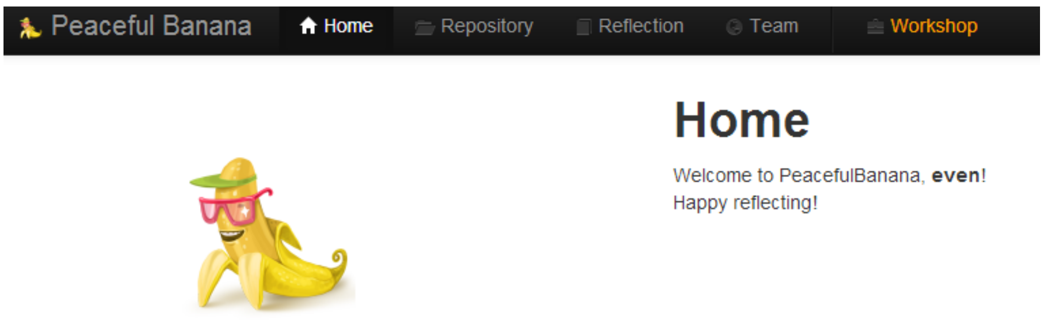
This chapter features the PeacefulBanana quick start guide, given to the students for our evaluation.
Peaceful Banana is a tool aimed towards aiding reflection in teams. It integrates with GitHub, collects the most relevant project artifacts and scaffolds these in order to trigger and promote reflection in your team. See you and your co-workers latest activity and use tag-cloud or statistics to create your daily reflection notes. PeacefulBanana allows you to choose what to share, and what to keep private! Reflect on your individual work by revisiting reflection notes, or reflect on your team's activity through reflection workshops and much more.

A.1 Getting started

So you and your team are developing software, using an agile process model? In that case you are probably familiar with the aspect of reflection or retrospectives in agile methods. The purpose of these retrospectives after each iteration is mainly to learn what works and what does not work, in order to make adjustments for the next iteration. More specifically your team wants to answer two fundamental questions:

- What went well during the last iteration that we continue doing?
- What could we do differently to improve?

The PeacefulBanana tool can help you and your team to identify the common ground answers to these questions. But first, we need to ensure you have the tools needed to use PeacefulBanana:
In order to use the Peaceful Banana application you need access to a device with an up-to-date Internet browser, such as Firefox, Google Chrome, Safari, Opera or Internet Explorer(version 7 and up). Other browsers present on tablets and smartphones may also work, but we suggest using one of the mentioned here.
That's it! That is all you need to have in order to begin using PeacefulBanana. In order to access the web-application, you need to open your browser and access this url: `http://vm-6121.idi.ntnu.no:8080` . Here you will be greeted by our welcome screen



Home

Welcome to PeacefulBanana, **even**!
Happy reflecting!

Figure A.2: PeacefulBanana Landing Screen

A.1.1 Registration

In order to become part of the PeacefulBanana community, you need to register for a new account. To do this, click the login button in the top right corner. At the login screen press the *Not yet a user?* link, as shown here:

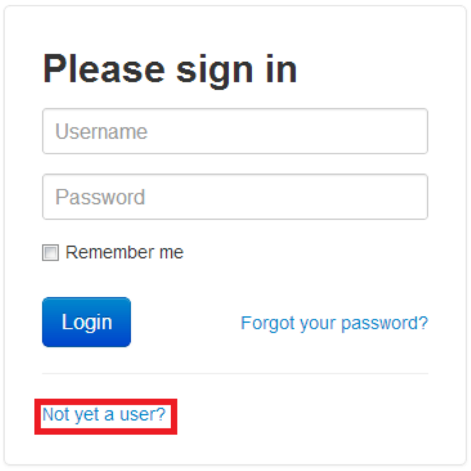
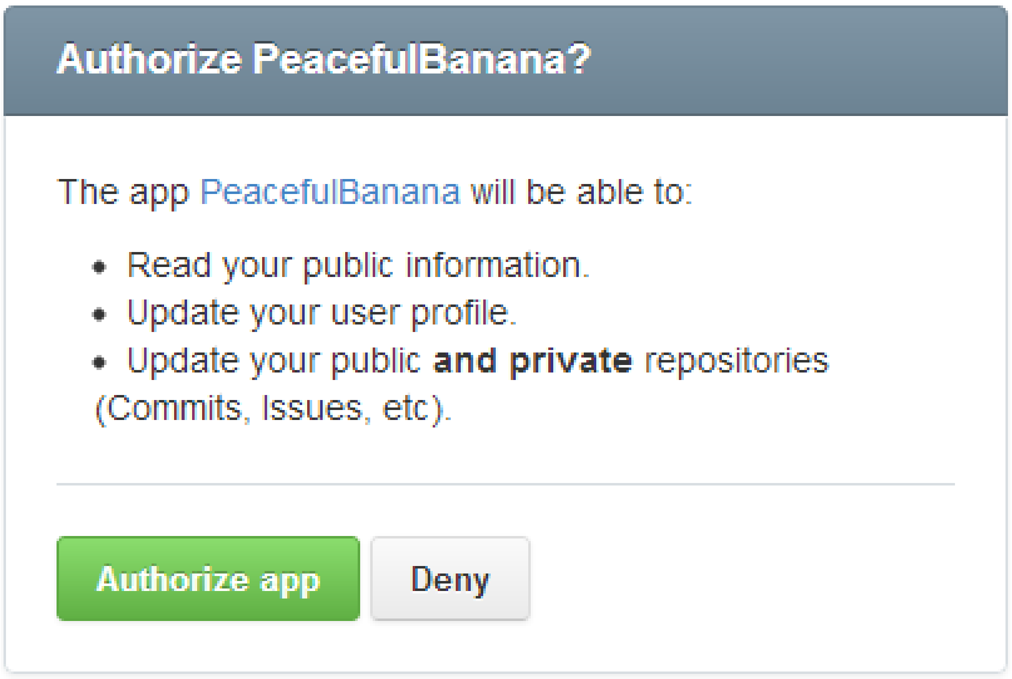


Figure A.3: PeacefulBanana Login screen

After filling in your registration details, you will receive an email containing a confirmation link. By clicking this link you will automatically be logged in to PeacefulBanana and redirected to GitHub for authorization. In order for PeacefulBanana to collect necessary data from GitHub, you will need to authorize our application, this will look something like this:

This registration and authorization is only a one-time process. When you have registered and authorized the PeacefulBanana application, you won't need to do think about it again. We'll take care of the rest.


Authorize PeacefulBanana?

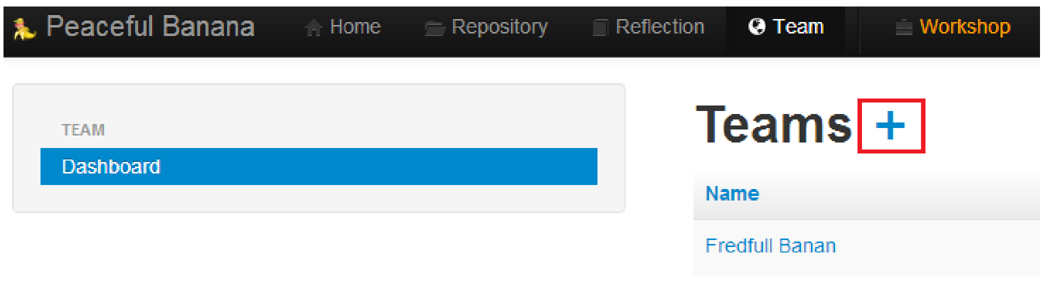
The app PeacefulBanana will be able to:

- Read your public information.
- Update your user profile.
- Update your public **and private** repositories (Commits, Issues, etc).

Figure A.4: PeacefulBanana - GitHub application authorization

A.1.2 Choosing team

Now you are ready to explore what PeacefulBanana can offer you and your team. The first screen you will see is the team screen:



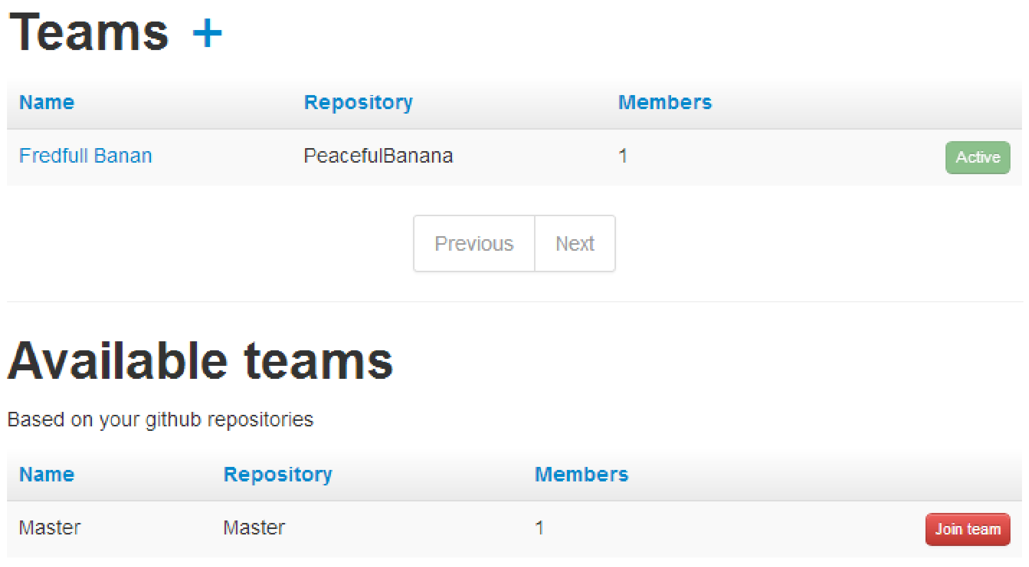
Name
Fredfull Banan

Figure A.5: PeacefulBanana - Team page, currently no teams have been created

The first thing you have to do is get the team leader to create a new team. One PeacefulBanana team retrieves data from your chosen GitHub repository and only that. All repository collaborators can join the PeacefulBanana team, so if you are having trouble joining the team - check if you have the collaborator status on the GitHub repository. Creating a team can be done simply by clicking the highlighted +

On the next screen, choose a suitable team name and which repository this team should be linked to. As stated earlier, there can only be one team for each repository, and only the team leader needs to create the team. After clicking *Create* PeacefulBanana will working it's magic and retrieve your chosen repositories' data. This includes commits, milestones, issues you are already familiar with on GitHub. This process might take a while the first time around, but don't worry, you will be notified when this process is complete.

Back at the team page, your team will be created and visible under *Teams*. Also the newly created team will be available to join for the other team members. Go ahead and invite your co-collaborators to join!


Teams +

Name	Repository	Members	
Fredfull Banan	PeacefulBanana	1	Active

Available teams

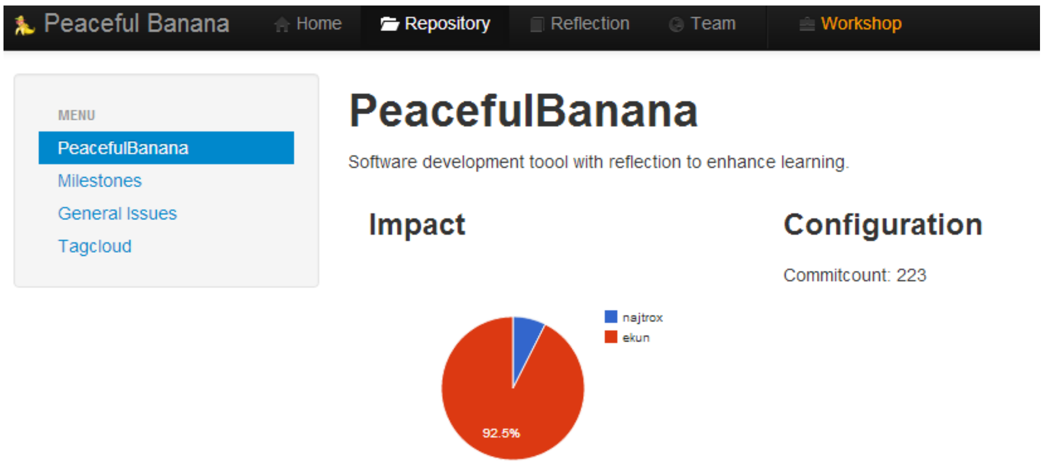
Based on your github repositories

Name	Repository	Members	
Master	Master	1	Join team

Figure A.6: PeacefulBanana - Team page showing current team and available teams based on your repositories

A.1.3 Repository tab

Clicking the repository tab, you will be presented with a screen like this: It's a simple start screen with the overall commit impact for your team's repository and a commit count. This information can be used to gain an



PeacefulBanana

Software development toool with reflection to enhance learning.

Impact

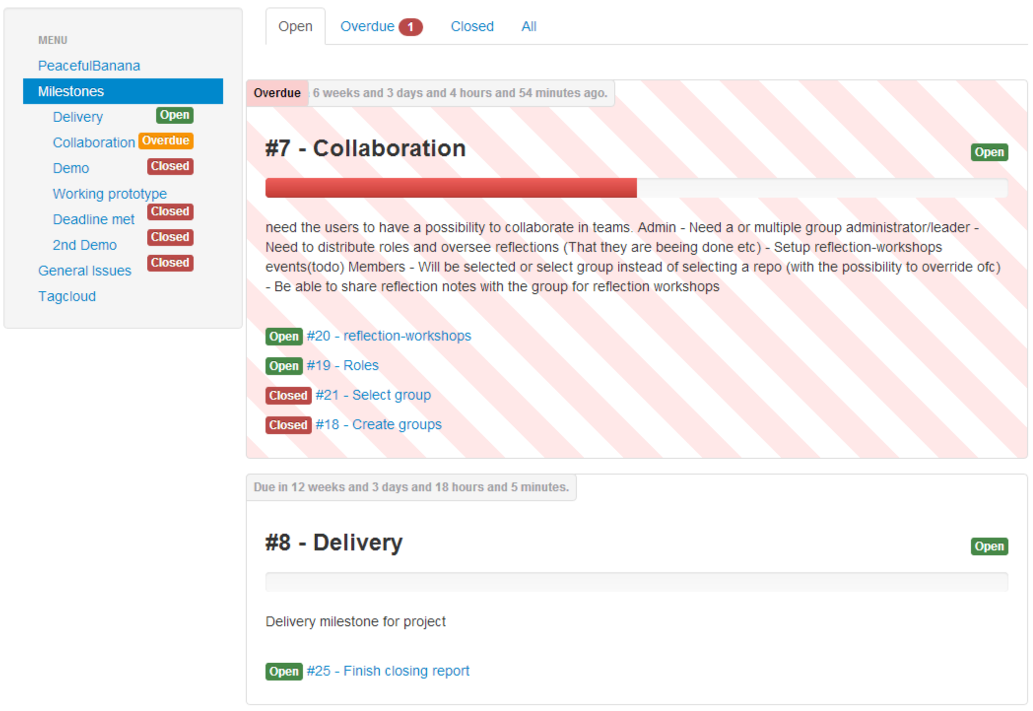
Configuration

Commitcount: 223

Figure A.7: PeacefulBanana - The main repository screen

indication of who is having the largest impact in the current team and how much overall activity there is. On the menu to the left you can see tabs for your repositories' milestones, issues and tag cloud.

The milestones sub-tab:



Overdue 6 weeks and 3 days and 4 hours and 54 minutes ago.

#7 - Collaboration Open

need the users to have a possibility to collaborate in teams. Admin - Need a or multiple group administrator/leader - Need to distribute roles and oversee reflections (That they are beeing done etc) - Setup reflection-workshops events(todo) Members - Will be selected or select group instead of selecting a repo (with the possibility to override ofc) - Be able to share reflection notes with the group for reflection workshops

Open #20 - reflection-workshops
Open #19 - Roles
Closed #21 - Select group
Closed #18 - Create groups

Due in 12 weeks and 3 days and 18 hours and 5 minutes.

#8 - Delivery Open

Delivery milestone for project

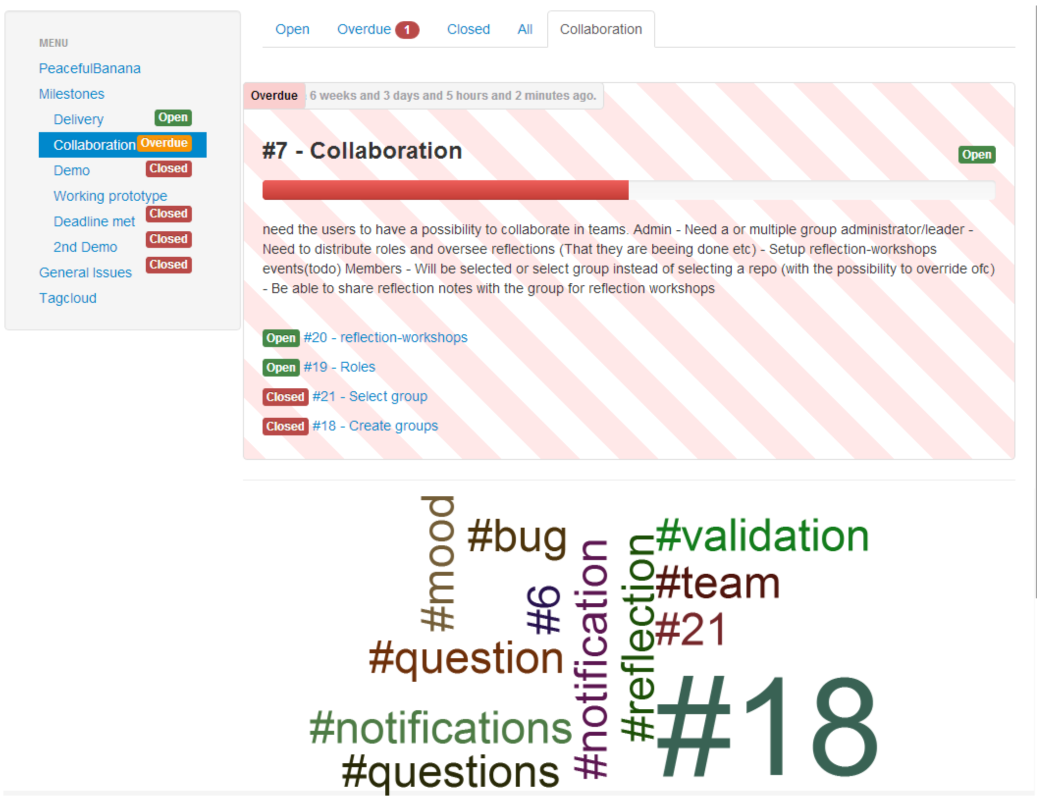
Open #25 - Finish closing report

Figure A.8: PeacefulBanana - All the repositories' milestones

On the milestones sub-menu you have access to all your milestones and their status(Open, Overdue or Closed). The main tab will by default show all your Open milestones. At the top you can switch to see milestones with other statuses.

It's fast and easy to switch between your currently open milestones, the ones that are overdue and the milestones you already have closed. This functionality makes it useful for individual daily usage, and for preparation to team retrospectives. Say your team missed an important deadline, and you wish to dive into what issues were causing problems. Clicking on this milestone will present you with all the issues that are connected to that milestone, in a clean manner: PeacefulBanana shows you that particular milestone's issues and their status, in addition to a milestone specific tag cloud. This tag cloud will help you identify which tags[1] are the most active, related to this particular milestone. This will in turn identify which issues might be worth taking a closer look at, while excluding the less important ones. Say if you see that issue #20 has been heavily referenced in commits, and you want to see further details surrounding that issue, simply click it:

[1]# perpended words in commit-messages



Overdue 6 weeks and 3 days and 5 hours and 2 minutes ago.

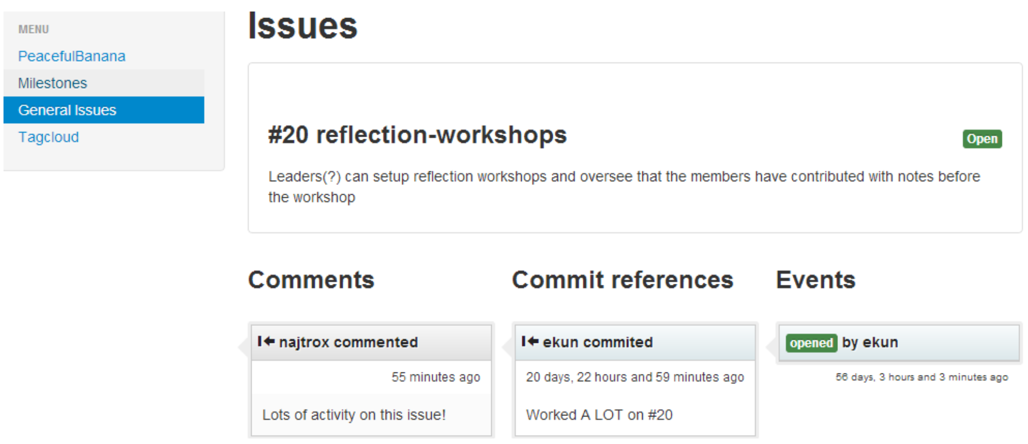
#7 - Collaboration Open

need the users to have a possibility to collaborate in teams. Admin - Need a or multiple group administrator/leader - Need to distribute roles and oversee reflections (That they are beeing done etc) - Setup reflection-workshops events(todo) Members - Will be selected or select group instead of selecting a repo (with the possibility to override ofc) - Be able to share reflection notes with the group for reflection workshops

Open #20 - reflection-workshops
Open #19 - Roles
Closed #21 - Select group
Closed #18 - Create groups

Figure A.9: PeacefulBanana - Single milestone

PeacefulBanana shows you this issue's comments, references and events all



Issues

#20 reflection-workshops Open

Leaders(?) can setup reflection workshops and oversee that the members have contributed with notes before the workshop

Comments

najtrox commented
55 minutes ago
Lots of activity on this issue!

Commit references

ekun commited
20 days, 22 hours and 59 minutes ago
Worked A LOT on #20

Events

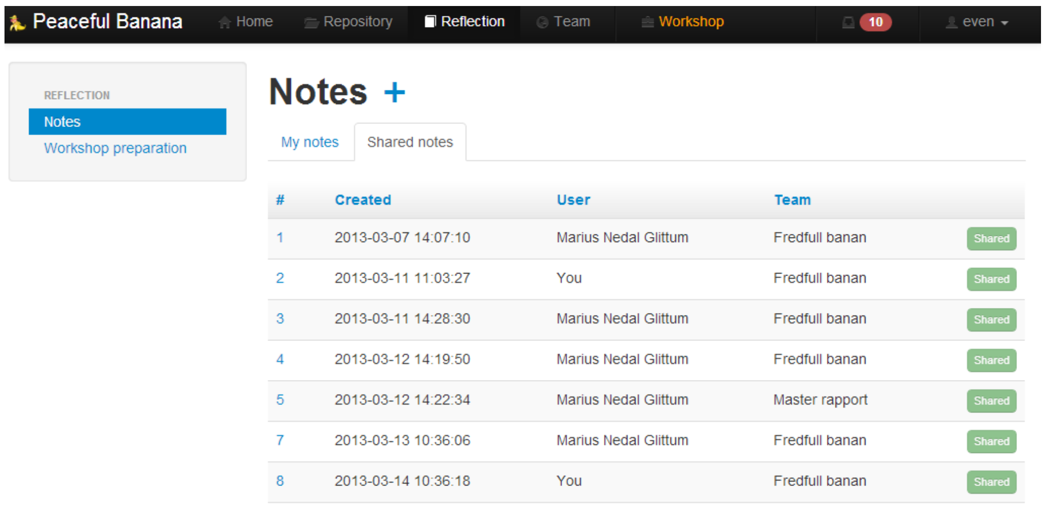
opened by ekun
56 days, 3 hours and 3 minutes ago

Figure A.10: PeacefulBanana - Issue #20, listing comments, references and events

in one clean interface, so that you or your team can quickly identify the activities.

A.1.4 Reflection & Reflection notes

A typical scenario in retrospectives, is that it's often hard to recollect all events for the last two-three weeks. PeacefulBanana aims to remove this issue completely from the retrospective sessions.
The reflection tab contains your personal reflection notes, and also your team's shared reflection notes. After creating a personal note, you can choose to share one or more of these with your team. It is not shared by default, so your personal notes really are personal. It is encouraged to share reflection notes with the team, the more notes shared - the easier it is to find common improvements and difficulties in the team. Personal notes may be used by you for personal reflection, or as preparation for a team retrospective meeting. The shared notes can be used by the team during the retrospective meeting.
PeacefulBanana also provides you and your team with a mood-graph under

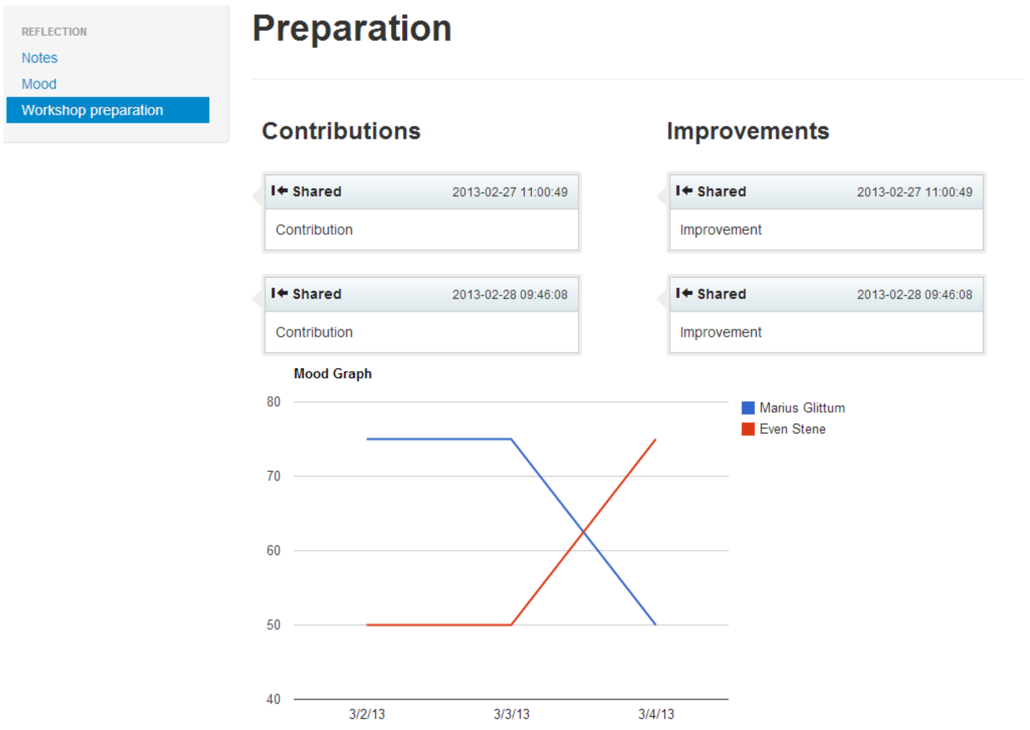


#	Created	User	Team	
1	2013-03-07 14:07:10	Marius Nedal Glittum	Fredfull banan	Shared
2	2013-03-11 11:03:27	You	Fredfull banan	Shared
3	2013-03-11 14:28:30	Marius Nedal Glittum	Fredfull banan	Shared
4	2013-03-12 14:19:50	Marius Nedal Glittum	Fredfull banan	Shared
5	2013-03-12 14:22:34	Marius Nedal Glittum	Master rapport	Shared
7	2013-03-13 10:36:06	Marius Nedal Glittum	Fredfull banan	Shared
8	2013-03-14 10:36:18	You	Fredfull banan	Shared

Figure A.11: PeacefulBanana - Reflection tab with the currently shared notes

the tab *Mood*, which ranges from members being very happy, to very sad. This mood graph will give you and your team an indication of how the individual moods progressed through the project. Is there any specific period where the team's mood ranged a lot? Then this period would be worth having a second look, as to what exactly was being worked on at the time, any specific problems or issues that could have been avoided?

Another feature of PeacefulBanana is the the tab *Workshop Preparation*, which helps users prepare for the team's retrospective sessions. You can easily switch between created workshops, and see your individual notes created for that particular workshops'. Revisiting these contributions and improvements helps users recollect the experiences from the particular days, the daily contributions and improvements, and finally that periods mood graph for both you and your team.

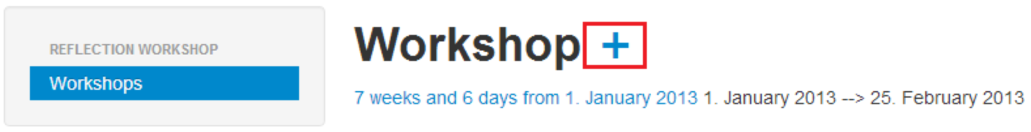


Contributions	Improvements
Shared 2013-02-27 11:00:49	Shared 2013-02-27 11:00:49
Contribution	Improvement
Shared 2013-02-28 09:46:08	Shared 2013-02-28 09:46:08
Contribution	Improvement

Figure A.12: PeacefulBanana - Reflection tab with the currently shared notes

A.1.5 Workshop tab

Finally we have the workshop tab. This is only visible if you are the manager of your current active team. This means that only the team leader can create workshops. If you wish to create a new workshop, simply click the + icon next to the title:



7 weeks and 6 days from 1. January 2013 1. January 2013 --> 25. February 2013

Figure A.13: PeacefulBanana - Create a new workshop

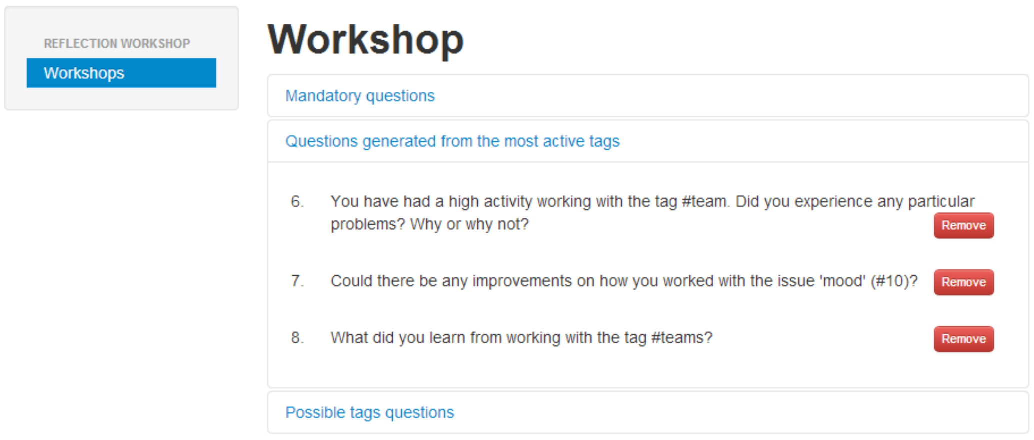
Which in turn will bring you to a date picker. Simply pick the from and to date you wish to set up a workshop on and click *Create*. After inspecting the newly created workshop you will see 3 tabs and some questions.

Mandatory questions

This tab is visible by default, and these questions are general for all workshops, and so they cannot be removed from the workshop. These questions are designed to help your team reflect on general issues related to team collaboration and improvement.

Questions generated from the most active tags

This tab features questions auto-generated and added by PeacefulBanana.



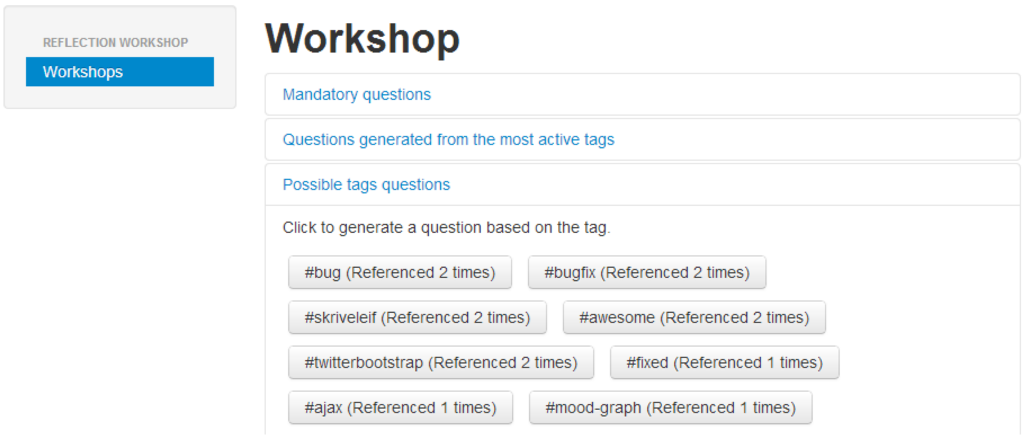
6. You have had a high activity working with the tag #team. Did you experience any particular problems? Why or why not?
7. Could there be any improvements on how you worked with the issue 'mood' (#10)?
8. What did you learn from working with the tag #teams?

Figure A.14: PeacefulBanana - Workshop questions related to the most active tags

The questions are generated based on the most active tags on your team for the period the workshop spans. If you do not wish to keep any of these questions simply click the *Remove* button. This will move the questions down to the *Possible tag questions* tab.

Possible tag questions

This tab contains all tags related to your teams current chosen workshop, in a descending order based on number of times the tag has been referenced in a commit. Clicking any of these tags will auto-generate a relevant question regarding this tag, and move it to the tab above. This allows you to quickly add and remove questions to your workshop.



Click to generate a question based on the tag.

#bug (Referenced 2 times) #bugfix (Referenced 2 times)
#skriveleif (Referenced 2 times) #awesome (Referenced 2 times)
#twitterbootstrap (Referenced 2 times) #fixed (Referenced 1 times)
#ajax (Referenced 1 times) #mood-graph (Referenced 1 times)

Figure A.15: PeacefulBanana - List of tags to generate questions from

A.1.6 Summary

1. Register in the PeacefulBanana application. Remember to check your email in order to complete the registration.
2. Set an active team in the *Team* tab. If there is no team, the team leader needs to create it.
3. Wait for the GitHub sync to complete, you will be notified when the process is completed. Be aware that this might take a while if this is your first time syncing with the server.

4. You are now ready to complete your daily notifications or explore the app on your own and with your team.
   Happy Reflecting

Useful Links

PeacefulBanana web-application: `http://goo.gl/JaUUg`

Appendix B

PeacefulBanana Database Descriptions

This part of the appendix will describe a detailed walkthrough of each class in PeacefulBanana.

B.1 User data

The first section is devoted to show and tell all data related to users, the tables described represents the different data types associated with users. Users are a vital part of the implementation and everything is centered around these users, it is therefore important to store as much relevant data as possible.

User

This table hold data related to each user and only the data which is unique to them. The password are encrypted with SHA-512 encryption which is a part of SHA-2 a set of encryption hashing functions created by the U.S. National Security Agency in 2002 and is a known not to be reversal.

Field	Description
Id	Unique id generated when the user is first created
UserName	This field stores the users username
Password	This field stores the users password as a encrypted string, the password is encrypted with SHA-512 encryption
Email	This field stores the users email
FirstName	This field stores the users first name
LastName	This field stores the users last name
SelectedRepo	The GitHub generated id of the repository currently selected by the user
GitLogin	The users GitHub login, used to bind commits to the user
DateCreated	This field stores the date when the user was first created

Role

The different roles that the users can be assigned to. These are hierarchy so that the highest gains access to the subset of this access level.

Field	Description
Id	Unique id generated when the role is first created.
Authority	This field contains the level of authority.

UserRole

Both fields in this table are combined for the primary key such that each role can only be given to each user once and each users can attain several roles.

Field	Description
User	Foreign key for the users id
Role	Foreign key for the role id

Notification

Notifications are used to prompt users about reflection sessions and such.

Field	Description
Id	Unique id generated when the notification is first created
Title	This field stores the title
Body	This field stores the entire message.
DateCreated	Time stamp generated when the notification is first created.
Unread	Boolean variable to state if the message is read.
Cleared	Boolean variable to state if the message is cleared or not.
NotificationType	The notification type, possible types are 'Reflection' or 'Other'.
User	A foreign key to the user that received the notification.

B.1.1 GitHub data

In this section we discuss the storage of data collected from GitHub. When we started development we tested with the data stored at GitHub, so we asked GitHub for the data every time it was needed, this made the application slow so we decided to store GitHub data locally to enhance performance. The following data-types where stored in tables as described below.

Milestones

Milestones are major goals consisting of several minor goals called issue, with or without a due date.

Field	Description
GithubId	Unique id from github.com
Name	The milestones name.
Description	A description of the milestone, like what features is to be implemented.
Status	A variable to say if its open or closed.
CreatedDate	The time stamp which the milestone is created.
DueDate	A date which the milestone is due on. This is null if the milestone does not got a due date.
ClosedDate	A time stamp when the milestone is closed.

Issues

Issues are minor goals or bugs, these issues can be bound to a milestone or independent.

Field	Description
GithubId	Unique id from github.com
Name	The issues name.
Body	A description of the milestone, like what features is to be implemented.
State	A variable to say if its open or closed.
Number	A number unique to the repository which is used for referring the issue in the commit messages.
Repository	A foreign key to the repository it is bound to.
MilestoneNumber	The milestone which the issue is bound to.
CreatedAt	The time stamp which the milestone is created.
UpdatedAt	The time stamp which the milestone last was updated.

Commits

At any given time a developer changes something in the source code it can be committed to the version control system, in this case git/github. These are the commits we store and use for reflection.

Field	Description
GithubId	Unique id from github.com
Message	The commit message which we gather tags from, these tags are marked hash tags.
Login	The GitHub user which did the commit.
CreatedDate	The time stamp which the commit is created.
Additions	A date which the milestone is due on. This is null if the milestone does not got a due date.
Deletions	A time stamp when the milestone is closed.
Total	A total of lines in the commit.

Repository

The current data is stored about each repository, in the corresponding fields in the table bellow.

Field	Description
GithubId	Unique id from github.com
Name	The repository name.
Description	A description of the repository.
CreatedAt	The time stamp which the commit is created.
UpdatedAt	The time stamp recorded when the milestone last was updated.

B.1.2 Collaboration data

Teams are bound uniquely to a repository, each repository will only have one team bound to it and every user with access[1] to the repository on GitHub will have the possibility to join the team.

[1]Being a collaborator of the project on GitHub.

Team

These teams of users are bound uniquely to a repository and there can only exists one team per repository.

Field	Description
Id	Unique id generated when the team is created.
Owner	A foreign key to the user that created the team.
Name	The teams name.
Repository	The githubId of the repository bound to the team.

TeamUser

This contains the users attached to each team and data relevant.

Field	Description
UserId	A foreign key to the users id.
TeamId	A foreign key to the teams id.
Role	The users role in the team. There are possible roles are 'Manager' and 'Developer'
Active	A variable that states if the user has selected this team as his active. Only one can be active per user at any given time.

B.1.3 Reflection data

Notes

These notes are bound to a team and there can only be created one for each team by a user each day.

Field	Description
Id	Unique id generated when the note is created.
Team	A foreign key to the team the note is bound to.
User	A foreign key to the user that created the note.
Mood	A level regarding the current mood the user that created the note recorded for that day. Mood is recorded as a number between 1-100 where 100 is very happy and 1 is very sad.
Contribution	The top 2 contributions done that day for the team by the user.
Improvements	The 2 things that can be improved by the user that day for the team.
Shared	Variable that states that the note is shared with the team.
CreatedAt	The time stamp which the commit is created.
UpdatedAt	The time stamp recorded when the milestone last was updated.

Workshop

Managers or owners of teams can create reflection workshops, where it is generated a set of questions based on the tags used by the teams members in the selected period.

Field	Description
Id	Unique id generated when the workshop is created.
TeamId	A foreign key to the team.
CreatedAt	The time stamp which the workshop is created.
DurationStart	The time stamp which the workshop period starts.
DateStart	The time stamp which the workshop period ends.

WorkshopQuestions

This table holds the questions for each workshop and indicates if they are mandatory or not.

Field	Description
Id	Unique id generated when the question is created.
Workshop	A foreign key to the workshop.
QuestionText	The generated question.
CommitTag	The tag which the question is generated from.

Appendix C

Usability Test-plan

Peaceful Banana

Usability Test Plan

Version: 1.0

Even Stene, Marius Nedal Glittum
23.01.2012

1 Table of Contents

2 Document Overview

This document describes a test plan for conducting a usability test during the development of *PeacefulBanana*. The goals of usability testing include establishing a baseline of user performance, establishing and validating user performance measures, and identifying potential design concerns to be addressed in order to improve the efficiency, usability, and end-user satisfaction.

The usability test objectives are:

- To determine design inconsistencies and usability problem areas within the user interface and content areas. Potential sources of error may include:
    - Navigation errors – failure to locate functions, excessive actions to complete a function, failure to follow recommended screen flow.
    - Presentation errors – failure to locate and properly act upon desired information in screens, selection errors due to labeling ambiguities.
    - Control usage problems – improper toolbar or entry field usage.
- Exercise the application or web site under controlled test conditions with representative users. Data will be used to access whether usability goals regarding an effective, efficient, and well-received user interface have been achieved.
- Establish baseline user performance and user-satisfaction levels of the user interface for future usability evaluations.

The PeacefulBanana application is developed with developers in mind, and will be evaluated on fellow students in the field of Computer Science. The testing will occur in a controlled environment in a private room.

3 Executive Summary

The *PeacefulBanana* tool is aimed towards aiding reflection for developers through multiple representations of relevant data. The tool integrates with Github, which features version-tracking support and several useful tools for version control. The tool provides an additional layer of features aimed towards aiding reflection in an agile development project.
The tool have many different possible scenarios, but we have we have identified two main scenarios:

Scenario 1 – Individual use on a daily basis
At the end of each work day, users entering the web-application will receive a notification with a request to do the daily notification. This daily notification consists with a summary of their individual activity the last 24 hours for the current active team. The user are prompted to input todays mood, their top 2 contributions and their top 2 points to improve on. These experiences can then be shared with the team, and is stored for later review.

Scenario 2 – Team use after each iteration
Users will be using the tool as part of the two-week reflection sessions that are present in many agile methodics. The team will use the tool to indicate how the project has been progressing over the last iteration, tightly coupled with one or several identified milestones. The tool gathers relevant data and presents it using multiple representations in order to revisit experiences and trigger reflection in the users. Examples of such data are tagclouds generated based on the most active issues in milestones, activity graphs and mood trajectories.
If any notes from scenario 1 has been shared, these will also be available to the team. This allows the team to get even more details surrounding certain issues in the iteration, and create a discussion around the experiences made by the team. Revisiting these experiences and comparing the views of the different members may trigger reflection and users learn from these experiences.

We conduct this usability test in order to answer several important questions, regarding these scenarios. Is the application easy to use, and can users achieve their goals in a timely manner?
Also, does the tool present data and trigger reflection for the user? Feedback from the usability test will further aid design and help identify problem areas that might cause problems for users. Since the participants are all computer science experts, and are familiar with reflection we hope to receive valuable input regarding these concepts.

4 Methodology

The usability test will be conducted on 5 participants. User-interaction with the PeacefulBanana tool will be done through an Internet Browser. We will have the users answer an entrance questionnaire, in order to collect demographic information. During the usability test we will take notes of the user’s problems and concerns. When the test is completed we will have participants come with suggestions on improvement. Finally participants will answer a SUS form, which consists of 10 questions designed to measure user satisfaction.

4.1 Participants

As mentioned we expect at least 5 participants. As the *PeacefulBanana* tool will be used with developers, which all have computer science background, participants will be fellow master students on the Computer Science field.
These participants will all have a background from Computer Science, and will be familiar with usability testing and also have experience with the notion of reflection.
The participants' responsibilities will be to attempt to complete a set of representative task scenarios presented to them in as efficient and timely a manner as possible, and to provide feedback regarding the usability and acceptability of the user interface. The participants will be directed to provide honest opinions regarding the usability of the application, and to participate in post-session subjective questionnaires and debriefing.

4.2 Procedure

Participants will take part in the usability test in a private room at the university. A computer with the *PeacefulBanana* web application and supporting software will be used in a typical working environment. The participant's interaction with the application will be monitored by the facilitator seated in the same room. In addition to the facilitator, notes will be taken by a member of the team.

The facilitator will brief the participants on the web application and instruct the participant that they are evaluating the application, rather than the facilitator evaluating the participant. Participants will sign an informed consent that acknowledges: the participation is voluntary, that participation can cease at any time, and that their privacy of identification will be kept safe. The facilitator will ask the participant if they have any questions.

Participants will complete a pretest demographic and background information questionnaire. The facilitator will explain that the amount of time taken to complete the test task will be measured and that exploratory behavior outside the task flow should not occur until after task completion. At the start of each task, the participant will read aloud the task description from the printed copy and begin the task. Time-on-task measurement begins when the participant starts the task.

The facilitator will instruct the participant to 'think aloud' so that the facilitator may observe and take notes of user behavior and user comments.

After all task scenarios are attempted, the participant will complete the post-test satisfaction questionnaire.

5 Roles

The roles involved in a usability test are as follows. An individual may play multiple roles and tests may not require all roles.

Facilitator

- Provides overview of study to participants
- Defines usability and purpose of usability testing to participants
- Assists in conduct of participant and observer debriefing sessions
- Responds to participant's requests for assistance

Test Observers

- Silent observer
- Takes notes of identified problems, concerns, coding bugs, and procedural errors.
- Serve as note takers.

Test Participants

- Provides overview of study to participants
- Defines usability and purpose of usability testing to participants
- Assists in conduct of participant and observer debriefing sessions
- Responds to participant's requests for assistance

5.1 Ethics

All persons involved with the usability test are required to adhere to the following ethical guidelines:

- The performance of any test participant must not be individually attributable. Individual participant's name should not be used in reference outside the testing session.
- A description of the participant's performance should not be reported to his or her manager.

6 Usability Tasks

The usability tasks were derived from test scenarios developed from user-stories, shortly introduced above. Due to the short time for which each participant will be available, the tasks are the most common and relatively complex of available functions. The tasks are identical for all participants in the study.

The application will be tested in a development environment and databases will be populated during use, and are not pre-populated. This will ensure a similar experience as to what the users get when they first use *PeacefulBanana.* The web application will run on a local computer, and not on a dedicated server as it will when deployed. This and the possible extra overhead from development mode, may impact performance slightly in a negative way.

Tasks:
Here are the most common and important tasks related to the two scenarios, the recruited experts will try to perform. These tasks are the most typical in the overall scope of tasks that the application will support.

Context:
PeacefulBanana is a tool intended to promote reflection and allow for revisiting and learning from previous experiences. *PeacefulBanana* integrates with and collects data from the version-control system *Git.*

Scenario 1 tasks:

Task 1: You start the application for the first time, and want to login, link your account with Github and set an active repository.

Task 2: View your notifications.
2.2 – Find the "Congratulations" notification and archive it. Find the archive and see if the notification was indeed archived.

Task 3: Find the "Reminder: Daily Reflection" note and perform the daily summary.
3.2 – Find a daily summary note and share it. Verify that is has indeed been shared.
3.3 – Find your mood graph

Scenario 2 tasks:

Task 4: Create a team with the name "Tuttifrutti" and your previously chosen repository.
4.2 – Find your created team and set it to active.
4.3 – Identify the members on your team and their role.

Task 5: Find all your repositories' milestones.
5.2 – Identify your overdue milestones.
5.4 – Find your repositories issues
5.4 – Find issue #17 . What is the status of this issue? When was it opened and when was it closed?

Task 6: Generate a tagcloud for your current chosen repository.
6.2 – Identify the most used *tag* for your team and yourself
6.3 - Generate a tagcloud for your current chosen repository.
6.4 – Find the commit impact for your repository.

7 Usability Metrics

Usability metrics refers to user performance measured against specific performance goals necessary to satisfy usability requirements. Scenario completion success rates, error rates, and subjective evaluations will be used. Time-to-completion of scenarios will also be collected.

7.1 Scenario Completion

Each scenario will require, or request, that the participant obtains or inputs specific data that would be used in course of a typical task. The scenario is completed when the participant indicates the scenario's goal has been obtained (whether successfully or unsuccessfully) or the participant requests and receives sufficient guidance as to warrant scoring the scenario as a critical error.

7.2 Critical Errors

Critical errors are deviations at completion from the targets of the scenario. Obtaining or otherwise reporting of the wrong data value due to participant workflow is a critical error. Participants may or may not be aware that the task goal is incorrect or incomplete.

Independent completion of the scenario is a universal goal; help obtained from the other usability test roles is cause to score the scenario a critical error. Critical errors can also be assigned when the participant initiates (or attempts to initiate) and action that will result in the goal state becoming unobtainable. In general, critical errors are unresolved errors during the process of completing the task or errors that produce an incorrect outcome.

7.3 Non-critical Errors

Non-critical errors are errors that are recovered from by the participant or, if not detected, do not result in processing problems or unexpected results. Although non-critical errors can be undetected by the participant, when they are detected they are generally frustrating to the participant.

These errors may be procedural, in which the participant does not complete a scenario in the most optimal means (e.g., excessive steps and keystrokes). These errors may also be errors of confusion (ex., initially selecting the wrong function, using a user-interface control incorrectly such as attempting to edit an un-editable field).

Noncritical errors can always be recovered from during the process of completing the scenario. Exploratory behavior, such as opening the wrong menu while searching for a function, will be coded as a non-critical error.

7.4 Subjective Evaluations

Subjective evaluations regarding ease of use and satisfaction will be collected via questionnaires, and during debriefing at the conclusion of the session. The questionnaires will utilize free-form responses and rating scales.

7.5 Scenario Completion Time (time on task)

The time to complete each scenario, not including subjective evaluation durations, will be recorded.

7 Usability Goals

The next section describes the usability goals for *PeacefulBanana*

7.1 Completion Rate

Completion rate is the percentage of test participants who successfully complete the task without critical errors. A critical error is defined as an error that results in an incorrect or incomplete outcome. In other words, the completion rate represents the percentage of participants who, when they are finished with the specified task, have an "output" that is correct. Note: If a participant requires assistance in order to achieve a correct output then the task will be scored as a critical error and the overall completion rate for the task will be affected.

A completion rate of 100% is the goal for each task in this usability test.

7.2 Error-free rate

Error-free rate is the percentage of test participants who complete the task without any errors (critical **or** non-critical errors). A non-critical error is an error that would not have an impact on the final output of the task but would result in the task being completed less efficiently.

An error-free rate of 80% is the goal for each task in this usability test.

7.3 Time on Task (TOT)

The time to complete a scenario is referred to as "time on task". It is measured from the time the person begins the scenario to the time he/she signals completion.

7.4 Subjective Measures

Subjective opinions about specific tasks, time to perform each task, features, and functionality will be surveyed. At the end of the test, participants will rate their satisfaction with the overall system. Combined with the interview/debriefing session, these data are used to assess attitudes of the participants.

8 Problem Severity

To prioritize recommendations, a method of problem severity classification will be used in the analysis of the data collected during evaluation activities. The approach treats problem severity as a combination of two factors - the impact of the problem and the frequency of users experiencing the problem during the evaluation.

8.1 Impact

Impact is the ranking of the consequences of the problem by defining the level of impact that the problem has on successful task completion. There are three levels of impact:

- High - prevents the user from completing the task (critical error)
- Moderate - causes user difficulty but the task can be completed (non-critical error)
- Low - minor problems that do not significantly affect the task completion (non-critical error)

8.2 Frequency

Frequency is the percentage of participants who experience the problem when working on a task.

- High: 40% or more of the participants experience the problem
- Moderate: 20% - 39% of participants experience the problem
- Low: 20% or fewer of the participants experience the problem

8.1 Problem Severity Classification

The identified severity for each problem implies a general reward for resolving it, and a general risk for not addressing it, in the current release.

Severity 1 - High impact problems that often prevent a user from correctly completing a task. Reward for resolution is reduced redevelopment costs.

Severity 2 - Moderate to high frequency problems with moderate to low impact are typical of erroneous actions that the participant recognizes needs to be undone. Reward for resolution is typically exhibited in reduced time on task.

Severity 3 - Either moderate problems with low frequency or low problems with moderate frequency; these are minor annoyance problems faced by a number of participants. Reward for resolution is typically exhibited in reduced time on task and increased data integrity.

Severity 4 - Low impact problems faced by few participants; there is low risk to not resolving these problems. Reward for resolution is typically exhibited in increased user satisfaction.

9 Reporting Results

The Usability Test Report will be provided at the conclusion of the usability test. It will consist of a report and/or a presentation of the results; evaluate the usability metrics against the pre-approved goals, subjective evaluations, and specific usability problems and recommendations for resolution..

System Usability Scale

PeacefulBanana

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Appendix D

Usability Consent form

Usability test consent form

I hereby grant full permission to the **PeacefulBanana** team to take notes of my comments during the usability test for **PeacefulBanana** Web Application.

I understand that other NTNU employees involved with **PeacefulBanana** may review these usability notes.

In this usability test :

- You will be asked to perform certain tasks on the PeacefulBanana website.
- We will also conduct an interview with you.
- You will be asked to fill in a questionnaire form.

Participation in this usability study is voluntary. All information will remain strictly confidential. The descriptions and findings may be used to help improve the PeacefulBanana application. However, at no time will your name or any other identification be used. You can withdraw your consent and stop participation at any time.

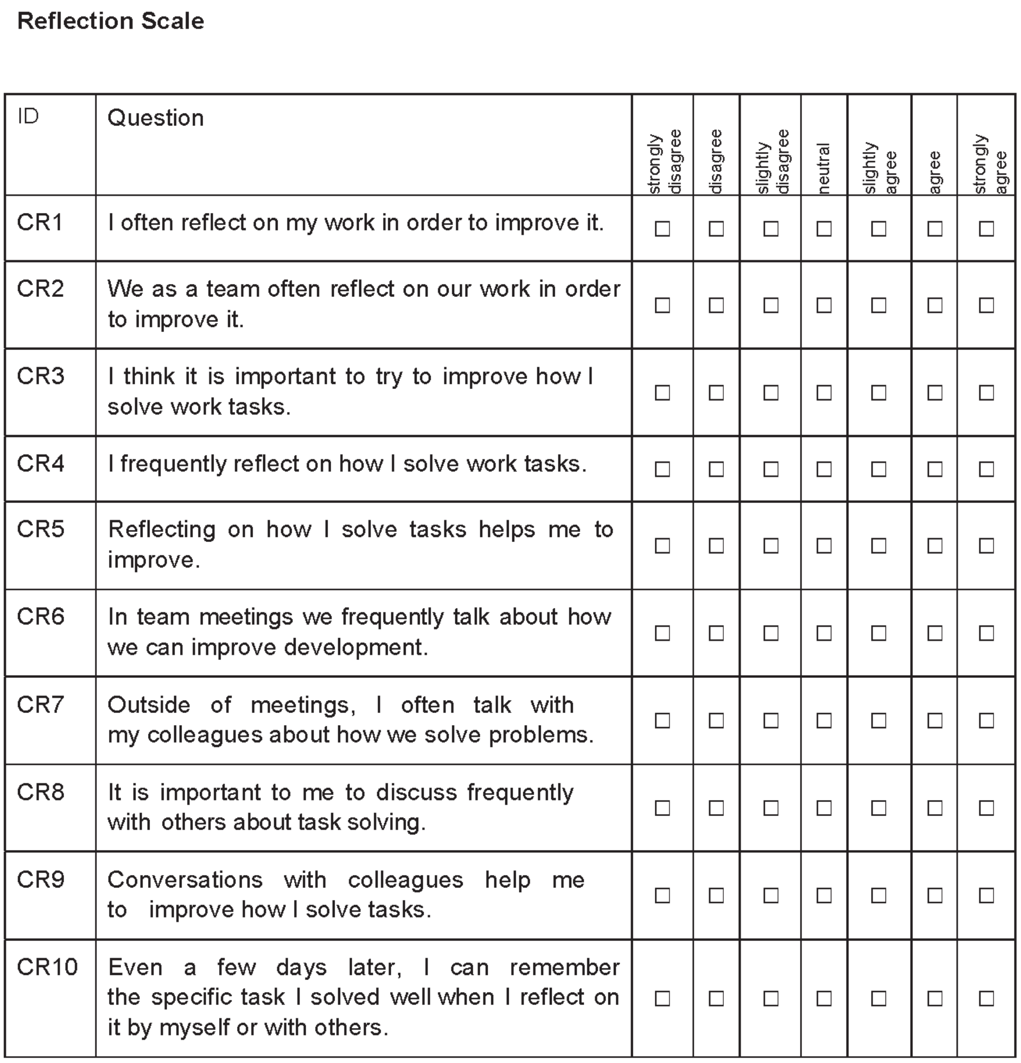
If you have any questions after today, please contact *Even Stene* at *XXXXXXXX*.

I have read and understood the information on this form and had all of my questions answered

Subject's Signature	Date
Usability Consultant	Date

Appendix E

Mirror-project Toolbox: Reflection scale



Reflection Scale

ID	Question	strongly disagree	disagree	slightly disagree	neutral	slightly agree	agree	strongly agree
CR1	I often reflect on my work in order to improve it.	☐	☐	☐	☐	☐	☐	☐
CR2	We as a team often reflect on our work in order to improve it.	☐	☐	☐	☐	☐	☐	☐
CR3	I think it is important to try to improve how I solve work tasks.	☐	☐	☐	☐	☐	☐	☐
CR4	I frequently reflect on how I solve work tasks.	☐	☐	☐	☐	☐	☐	☐
CR5	Reflecting on how I solve tasks helps me to improve.	☐	☐	☐	☐	☐	☐	☐
CR6	In team meetings we frequently talk about how we can improve development.	☐	☐	☐	☐	☐	☐	☐
CR7	Outside of meetings, I often talk with my colleagues about how we solve problems.	☐	☐	☐	☐	☐	☐	☐
CR8	It is important to me to discuss frequently with others about task solving.	☐	☐	☐	☐	☐	☐	☐
CR9	Conversations with colleagues help me to improve how I solve tasks.	☐	☐	☐	☐	☐	☐	☐
CR10	Even a few days later, I can remember the specific task I solved well when I reflect on it by myself or with others.	☐	☐	☐	☐	☐	☐	☐

Figure E.1: Reflection Scale from the MIRROR evaluation toolbox

Appendix F

Design-Science Research Guidelines

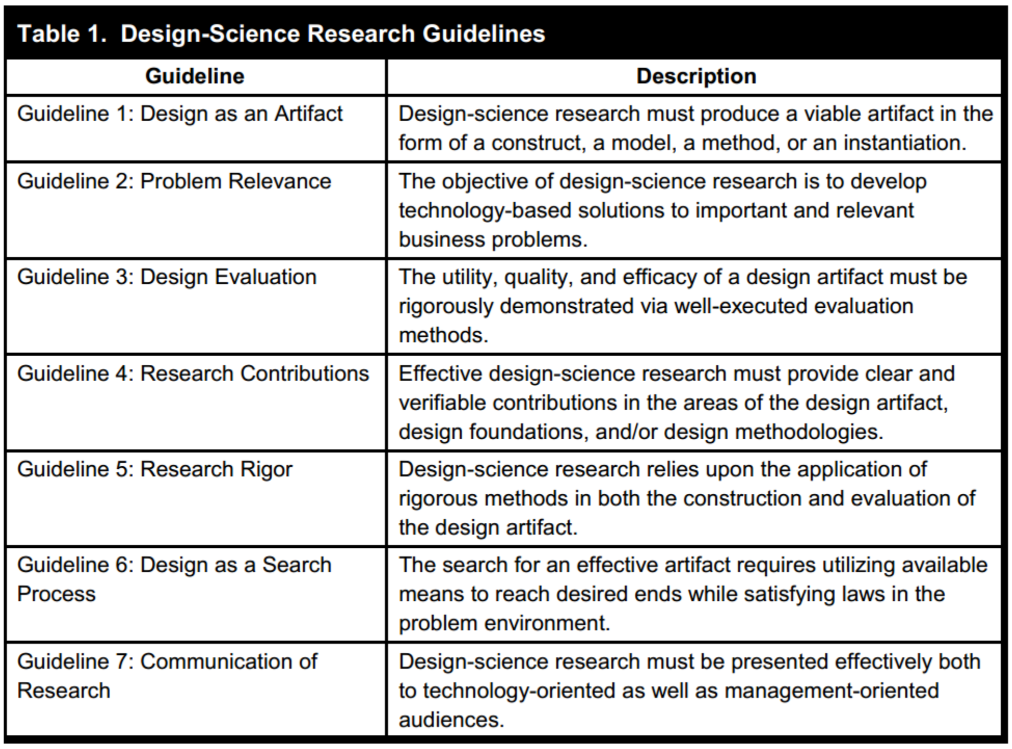


Table 1. Design-Science Research Guidelines

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Figure F.1: Design-Science Research Guidelines