Getting an overview of software development projects more efficiently

Simon Sandström

Simon Sandström
Spring 2013
Examensarbete, 15 hp
Supervisor: Marie Nordström
External Supervisor: Linus Mähler Lundgren
Examiner: Pedher Johansson
Kandidatprogrammet i datavetenskap, 180 hp

Abstract

At a software development company, many kinds of tools is often used to manage projects and other types of resources. One task that is performed many times every day by project leaders and developers at these companies is to get an overview of active projects and software in development. This report will examine if a new tool can be introduced that will make it more efficient to complete this task.

A specific company's procedure of performing this task including the tools used in this procedure have been examined. To improve it, an already existing tool was analysed but did not meet the stakeholders requirement. Instead a new tool was designed and a prototype was implemented, based on the analysis of the existing tool.

The result is a fully functional tool that can be used to get an overview of active projects, instead of using two different tools individually. Some initial usability tests also suggests that using this tool is more efficient. Although, it also shows that some information about projects can be harder to retrieve when using the new tool.

Acknowledgements

I wish to thank Linus Mähler Lundgren for giving me the opportunity to do this project and for guiding me through it and Marie Nordström for helping me and steering me in the right direction throughout the project.

I also want to thank Madelene Holmgren for giving me the energy needed to complete this project as well as reading the report and giving me valuable feedback.

Last but not least, I want to thank everyone at Dohi Sweden for making me feel welcome at your office and for helping me with the project in one way or another.

Thank you.

Contents

1 Introduction

Many kinds of different tools are used within software development companies. Tools for managing time, projects and other resources. Tools that are used for completing tasks in the software development process, such as writing, testing and deploying software.

The task of using these tools to get an overview of active projects needs to be performed many times every day in a software development company. This is an important task since an overview of project is needed for further planning of projects as well as to identify projects that are going less well and needs attention. As more new tools are being used, and tools being replaced, this task can grow complex and time consuming.

1.1 Background

Dohi Sweden is a holding company that offers products and services in the audiovisual industry. To manage projects in their daily work they use a tool, Pivotal Tracker, which is a web-based agile project management tool[1]. Another tool that is used by the software developers at Dohi Sweden is Jenkins, which is a continuous integration server that monitors the execution of jobs[1][2]. At Dohi Sweden this tool is used to automatically build and test software when a developer commits code to the shared code repository.

To get an overview of the status of projects and software in development, a software developer or project leader must access both of the tools described above, using their respective web interface. This task includes, for both of these tools, logging in to the system, selecting the specific project or job and finally find the relevant information. Information that can be relevant to both software developers and project leaders includes how much work that is completed, what and when the next milestones is in specific projects in Pivotal Tracker. It also includes the execution status of specific jobs in Jenkins. Throughout this report the procedure of getting an overview of active projects using these two tools individually will be referenced to as the ”reference procedure”.

The idea behind this project is that it should be possible to get a complete overview of active projects by using a single tool instead of using every tool individually. This is illustrated in Figure 1. By using a this single tool time can be saved and spent on other tasks, and projects going less well can be identified and corrected earlier. In this report, the use of this tool to get an overview of active projects will be called the ”new procedure”.

[1] These jobs can be any kind of repeatable tasks that is run on a computer, such as compiling software, testing software, backing up files, and so on.

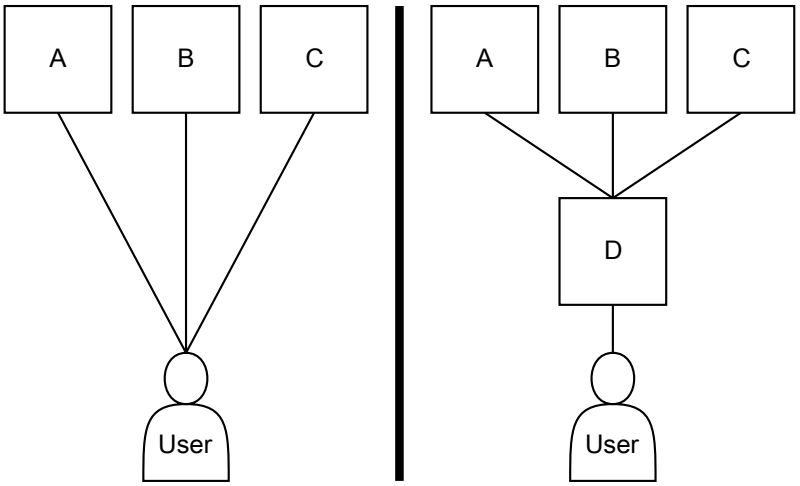

Figure 1: The user to the left is using all three tools A, B and C individually, while the user to the right only uses tool D, both for the purpose of getting an overview of active projects.

There are some requirements for a new tool. These was developed as a pre-study for this project and any details how the requirements was developed will not be presented in this report. The requirements are:

- **R1**: It must be able to gather and display some information from atleast the two tools; Pivotal Tracker and Jenkins.
- **R2**: The tool must be free to use.
- **R3**: The tool must be web-based such as it can be accessed from a number of different devices and operating systems.
- **R4**: The tool must be designed in a way that makes it more efficient to use for the purpose of getting an overview of active projects than by using the different tools individually.
- **R5**: The tool must support authentication for different users.
- **R6**: It must be possible to choose what kind of information that should be displayed for each user.
- **R7**: The tool must support plug-ins for retrieving and displaying information from other tools. This means that requirement **R1** must be met by the use of two plug-ins.

1.2 Goals

The goal of this project is to develop a new tool as described in the previous section. The goal can be divided in to these sub-goals:

1. Find and evaluate any existing tools that might meet the requirements of a new tool.
2. If no existing tool meets the requirements: design and implement a prototype of a new tool.
3. Compare the reference procedure with the new procedure, which uses the new tool, in the aspect of usability.

1.3 Development Method

During this project, an agile development method called Scrum[3] have been used. The general idea of Scrum is to divide the project into *iterations*. Each iteration, or *sprint*, is an abstract notion of a predefined time span with a clear and decisive goal. These sprints usually span over one to a few weeks. The purpose of having short sprints, and using agile development methods in general, is to quickly build prototypes that can be tested to make decisions on how to proceed.

Here follows a breakdown of typical daily and weekly activities that can be performed when using Scrum:

- Every *day* begins with a SCRUM meeting, discussing:
  - What was done yesterday
  - Problems that have been discovered
  - What is to be done until tomorrow
- Every *week* there is two bigger SCRUM meetings:
  - **Monday** - Plan and set the goals of the week
  - **Friday** - Summarise the week's work, successes and problems

1.4 Outline

Chapter 2 presents the reference procedure and describes it in detail. A list of the tools used in this procedure is presented including what the tools are, how they are used and why they are used. Chapter 3 presents an analysis of an existing tool that could be used in a new procedure to get an overview of projects. In chapter 4, the system design and implementation of a new tool, the prototype, is presented. Chapter 5 explains software usability and how describes how one can evaluate it in a software. It concludes with an description of how the usability tests on the new tool was performed as well as a summary of the results. Finally, in Chapter 6, a discussion of the results and the overall work of the project is presented.

2 Description of the Reference Procedure

There are two essential tools that are being used in the reference procedure for the purpose of getting an overview of active projects. These two tools will be described in detail, including what they are used for, how they are used and any technical details that can be of use when designing a new tool.

2.1 Pivotal Tracker

Pivotal Tracker is a web-based agile project management tool[1]. It is developed by Pivotal Labs and hosted on their web servers, which means that a user needs an Internet connection in order to use the tool. There are a few pricing plans to pick from depending on how many projects the user wants to have, how many collaborators all projects can have, and more.

The tool have a strong connection to the development method Scrum. The project is divided into iterations that can be of any length in terms of weeks. Each project have a current-iteration log and a product backlog that consists of user stories that can be of four different types:

Features: are stories that provide verifiable business value to the team's customer. These user stories is estimated in terms of points, which is described below.

Bugs: represents unintended behaviour of the product that must be fixed.

Chores: are stories that are necessary, but provide no direct, obvious value to the customer.

Releases: are milestone markers, and allow your team to track progress towards concrete goals.

The work needed to complete a *feature* is estimated in terms of points, which also can be seen as time. The tool can calculate approximately how much work is completed for each iteration and this approximation is used to fill the current-iteration log with user stories from the product backlog when a new iteration starts. This information is also used to show different types of charts that can be of use for the project team.

Pivotal Tracker is being used to manage all projects. It is used by developers to plan the projects that they are working on, adding features that must be completed, bugs that must be fixed and milestones in the development. Project leaders also use this tool to keep track of all the active projects that they are responsible for. They use the tool to check if milestones will be completed in time, to find projects that are going less well and needs attention, and much more.

Technical Details

Pivotal Tracker provides an API[1] that can be used to retrieve, create, update and delete information such as projects, iterations and stories, using the four HTTP[2] methods GET, POST, UPDATE and DELETE, respectively[4]. The response from the API for each request is formatted using XML[3]. To use the API, a user must authenticate himself using either their user name and password or using a user token that can be created after logging in on the Pivotal Tracker web site. Appendix B contains an example of a request and the response when using the API.

It is also possible to use the activity webhook[5] in Pivotal Tracker. Using this, Pivotal Tracker will send an XML formatted document using the HTTP POST request to any specified URL when there is activity for a specific project. The document contains detailed information about the activity, see AppendixB for examples.

2.2 Jenkins

Jenkins is a web-based tool used to monitor the execution of jobs[2]. As opposed to Pivotal Tracker, Jenkins can be run locally on the users own computer or on one of the company's server. Jenkins report results of each execution for each job, this report includes if the execution was successful or if it failed and the time taken for the execution.

The practice to continuously build and test software when new or changed source code have been committed to the shared code repository is called continuous integration[6]. Jenkins can be used as a continuous integration system by setting up jobs for each software project that automatically builds the software and executes unit tests when code have been committed to the repository.

Jenkins is being used as a continuous integration system. All of the software projects have a corresponding job in Jenkins. The information that Jenkins reports whenever a job has been executed, which is done when a developer commits new or changed code to the repository, is valuable for each of the members in the project including the project leader.

Technical Details

Jenkins have support for third party plug-ins, which means that new functionality can be added to the tool. There is many available plug-ins to use and most of these plug-ins are open source and free to use and modify. One of the plug-ins is *Notification Plugin*[4]. This plug-in uses the technique webhooks to send information about every execution of a specific job to a specified URL. The information is encoded using JSON[5] and contains information such as, the name of the job, the id of the execution, if the execution was started, completed or finished and the status if the execution finished. Examples of information that is sent is presented in Appendix B.

[1] Application Programming Interface: Describes how applications can communicate with each other.
[2] Hypertext Transfer Protocol: A protocol to transfer hypertext and other kinds of data between computers.
[3] XML is a language for encoding information.
[4] Official website:`https://wiki.jenkins-ci.org/display/JENKINS/Notification+Plugin`
[5] JavaScript Object Notation: A text based standard to encode data.

There is also an API that Jenkins provides. The information that the API provides can be retrieved in three different formats: XML, JSON and a Python-specific format. The Python-specific format can be converted directly to Python objects using the method *eval*[6]. Information such as: configured jobs, status of executed jobs and jobs in the execution queue can be retrieved with this API.

2.3 Conclusion

The two tools described in this chapter are used for different things. Pivotal Tracker is used to manage projects, which includes time, resources and milestones in the development of the software. Jenkins is used only in the development of the software to automate certain steps and to verify that the software does what it is supposed to do, without errors.

Both tools are web-based which means that can be accessed with the use of different devices and operating systems. It also means that the tools can be accessed anywhere, if the user have an Internet connection. A new tool, that a user should be able to use instead of using Pivotal Tracker and Jenkins individually, should therefore also be web-based and accessible via the Internet.

Another aspect is the amount of information that can be gathered using these tools. If a new tool should retrieve and collect information from these tools, the information must be filtered. Either the tool itself should be designed such as it only show some information, or so that each different user can decide what information should be displayed.

Both tools offers multiple ways of retrieving the information programmatically. This makes it easy for a new tool to be able to keep up to date information from these tools, fully automatically.

[6]See: `http://docs.python.org/2/library/functions.html#eval`

3 Analysis of an Existing Tool

One existing tool have been analysed to check whether it fits the requirements of a new tool. The analysis was also done in order to find architectural patterns, design patterns and other general design ideas that can be of use if the results show that this tool does not meet all the requirements.

3.1 ProjectMonitor

ProjectMonitor is a tool that aggregates information from continuous integration systems and displays it as a web page[7]. The tool, as of May 2013, supports six different continuous integration systems; *Cruise Control*[1], *Jenkins*, *TeamCity*[2], *Travis CI*[3], *tddium*[4] and *Semaphore*[5]. The purpose of this tool is to display job execution results from any of the supported systems on a single web page, which can be shown on a big screen monitor or TV, making it easy for developers to get an overview of active projects that they are involved in. ProjectMonitor is open source and licensed under the MIT license[6].

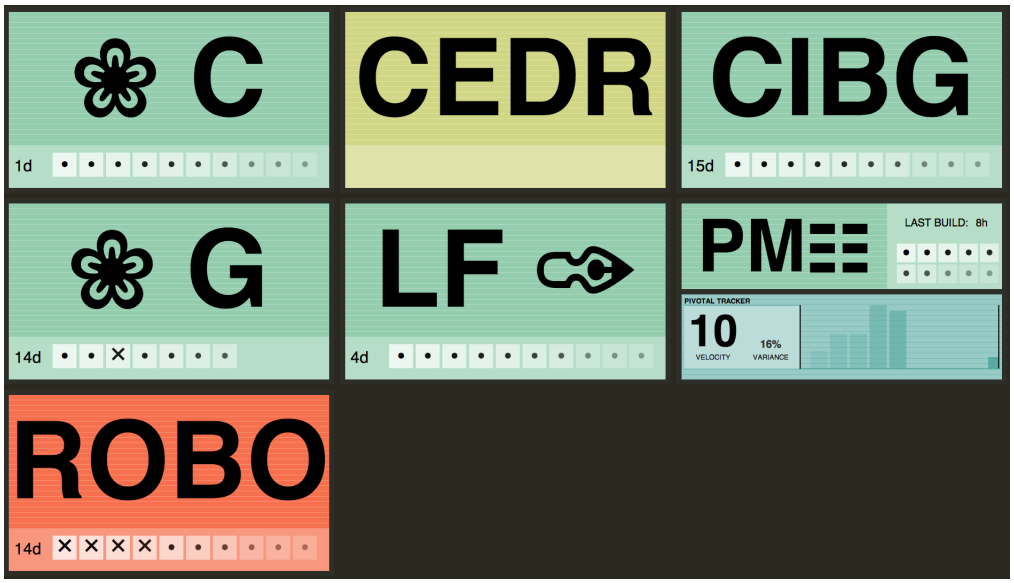

Figure 2: The web page that ProjectMonitor produces, with seven projects configured .

[1] http://cruisecontrol.sourceforge.net/
[2] http://www.jetbrains.com/teamcity/
[3] https://travis-ci.org/
[4] https://www.tddium.com/
[5] https://semaphoreapp.com/
[6] See https://github.com/pivotal/projectmonitor/blob/master/MIT.LICENSE

Figure 2 shows the web page that ProjectMonitor produces with seven projects configured. Each project is shown as a box with an abbreviation of its name, the time since the last execution of the job and the status for the last ten executions. A successfully executed job is denoted with a dot and an executed job that failed is denoted with a cross. The background color of the box also tell the user the statuses of the last executed jobs. A green background means that most of the executions was successful, a red background means that most of the executions was unsuccessful and a yellow background means that no executions have yet been completed.

Information are retrieved from the different continuous integration systems using one of two different techniques, depending on what the particular system supports:

Polling: ProjectMonitor polls the continuous integration system for new information periodically. The information is retrieved using the continuous integration systems API with the specified project id and login credentials.

Webhooks: The continuous integration systems is configured to send job status information to ProjectMonitor with the use of HTTP when certain events have occurred.

ProjectMonitor have an administration web interface so that an authenticated administrator can configure what jobs should be displayed. It does support Pivotal Tracker in the sense that a continuous integration system job can be associated with a project in Pivotal Tracker. The information shown about the project is limited and since a project in Pivotal Tracker must be associated with a job from one of the continuous integration systems, only projects with software being developed can be displayed.

3.2 Conclusion

ProjectMonitor does not fit all of the requirements of a new tool. The tool can only show information from a Pivotal Tracker project if there is an associated job in Jenkins. The stakeholders have many projects that does not have an associated job in Jenkins and projects that does not include developing software at all. This could be resolved by setting up jobs in Jenkins for each project in Pivotal Tracker that does not already have one. But it will take up both processing power and disk space, which is a disadvantage.

The other feature that ProjectMonitor lacks is the possibility for a specific user to only get information from a subset of the configured projects. For example: there are four active projects, project A, B, C and D. One user only wants to see information from project A and B while another user only wants to see information about project C and D. The project leader for all of these projects wants to see information from all four. As this tool is open source, this feature could have been added. To do this, one must read the source code of the software and learn how it is built and how it works. This can take a long time, especially if the source code is not very well commented. In this case the source code was not commented at all, so this approach was not feasible.

4 Prototype of a New Tool

Based on the requirements in Section 1.2, the theory behind Software Usability which is described in Chapter 5, the description of the reference procedure in Chapter 2 and the analysis of the existing tool in Chapter 3, a system design of a new tool is presented. A simple implementation of the system design is also presented in this Chapter.

4.1 System Design

The system design of the new tool was designed mainly based on the results of the analysis of the existing tool, including the architecture and design patterns used in this tool. The architectural pattern used is the client/server pattern, which in this case means that the prototype acts as a HTTP server and is accessible by a user with a client, such as a web browser. This makes it possible to access the tool from almost any type of device that is connected to the Internet. The two techniques polling and webhooks, which the existing tool that is described in Chapter 3 uses, are used in this system design. This makes it possible for the tool to automatically retrieve information from both Pivotal Tracker and Jenkins.

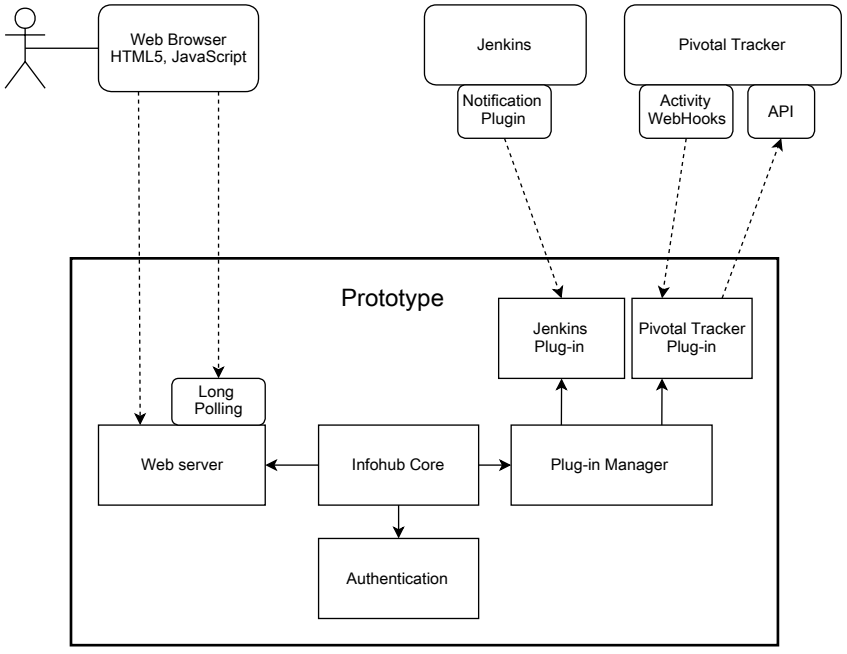

Figure 3: System design of the prototype. Continuous lines represents connections of the modules in the system and the dotted lines represents communication over a network connection, such as the Internet.

An overview of the system design is illustrated in Figure 3. The system can be divided into two parts which will be described individually. The first part, called *Web Server*, focuses on how a user interacts with the tool while the other part, called *Plug-Ins*, focuses on how the tool interacts and retrieves information from other systems. The module *Infohub Core* ties these two parts together and creates a fully functional tool that can be used to get an overview of active projects.

Web Server

This subsection describes the following modules: *Long Polling*, *Web Server* and *Authentication*, as illustrated in Figure 4.

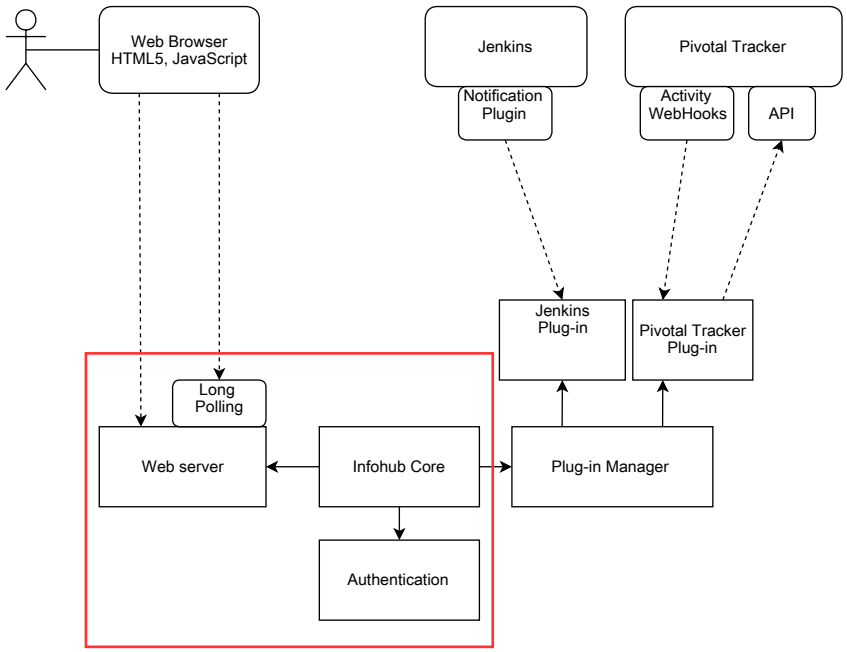

Figure 4: The modules Long Polling, Web Server, Infohub Core and Authentication.

To be able to access the tool from different places and different devices, a web server is used. When entering the web page that the web server produces, a user has to authenticate. The authentication is handled by the *Authentication* module, which verifies the users credentials and also tells the web server what types of plug-in specific settings the user have. The plug-ins and their settings is described in the next subsection. After the user have authenticated, a new web page is displayed with the information that is stored in the tool.

Long polling is used to dynamically update the information on the web page with new information from the tool. There are two techniques for polling[8]:

Short polling is the simplest of the two methods and is done by having the client periodically asks the server for new information. The server responds at once, with either that no new information is available or with the new information. As the client only polls for new information periodically, there can be a delay between when there is new information available and when the client actually gets the new information.

Long polling is a more complex method than short polling. It is also done by having the

client ask the server for new information. If there is new information to be returned, the server responds with the information at once. If there is no new information available, the server waits to respond until there actually is new information to be returned. It is almost no delay between when there is new information available and when the client gets the new information. This method uses more server resources than short polling due to having to keep the connections with the clients open as well as keeping track of connected clients. When new information have been received by the client it restarts the procedure.

Long polling was chosen because it minimises the delay between when new information is available and when the user actually sees the new information. Long polling does on the other hand increase the number of connections that the web server must keep track of. But since there will not be a large number of users connected at the same time, this will not cause any problems.

Plug-ins

This subsection describes the following modules: *Plug-in Manager*, *Pivotal Tracker Plug-in* and *Jenkins Plug-in*, as illustrated in Figure 5.

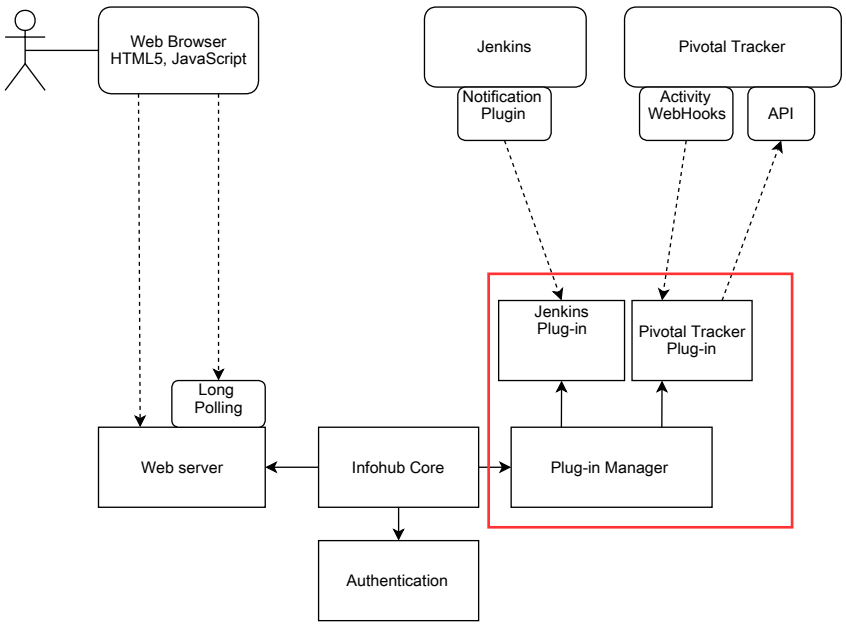

Figure 5: The modules Plug-in Manager, Pivotal Tracker Plug-in and Jenkins Plug-in

A plug-in system in a new tool is one of the requirements. This means that it should be possible to add new functionality in the aspect of supporting new systems without modifying many of the different modules in the system. A plug-in is a piece of software that adds functionality to another software, called the *host application* [9]. Plug-ins are used in this system design to handle all communication with other systems, such as Pivotal Tracker and Jenkins. The module *Plug-In Manager* is used to dynamically load and manage available plug-ins.

Each plug-in in this design have the following responsibility:

1. Return the stored information when the a user requests it via the web server. The user can have specific settings for this plug-in, which means that the stored information should be filtered before returned to the user. For example; the Pivotal Tracker plug-in have information stored for project A and B. The user have specific settings for this plug-in that says that the user only wants information from project A. Only information that belongs to project A should be returned.

2. Retrieve and store information. For example; the Pivotal Tracker plug-in should retrieve project information via the Pivotal Tracker API.

3. Provide a web page where a user can configure the plug-in specific settings.

This system design contains descriptions of two specific plug-ins; the Jenkins plug-in and the Pivotal Tracker plug-in. The Jenkins plug-in simply retrieves job notifications from the Jenkins server using webhooks, which is described in Section 2.1. A user can configure what jobs in Jenkins that should be displayed. The Pivotal Tracker plug-in uses two methods for retrieving information. The first method is to retrieve project activity information from Pivotal Tracker using webhooks and the other method is to retrieve project information from the Pivotal Tracker API. Both these methods are described in Section 2.2.

4.2 Implementation

The prototype that was implemented in this project is based on the system design that is presented in the previous section, but does not contain all the features in it. Figure 6 shows the modules that was implemented and the module were not. The following list summarises the features that is presented in the previous section and if they are implemented or not:

Implemented: Web server that serves an HTML document with JavaScript code that dynamically refreshes the content of the page with the technique long polling.

Implemented: A plug-in architecture. Uses an abstract class that declares three methods that a plug-in can override to add functionality to the prototype.

Implemented: Two plug-ins. One for Pivotal Tracker project information and one for Jenkins job execution statuses.

Implemented: Plug-in manager that initialises and uses all plug-ins that are available .

Not implemented: User authentication and user-specific settings for plug-ins. This means that all projects in Pivotal Tracker and all jobs in Jenkins that are configured will be displayed for all users.

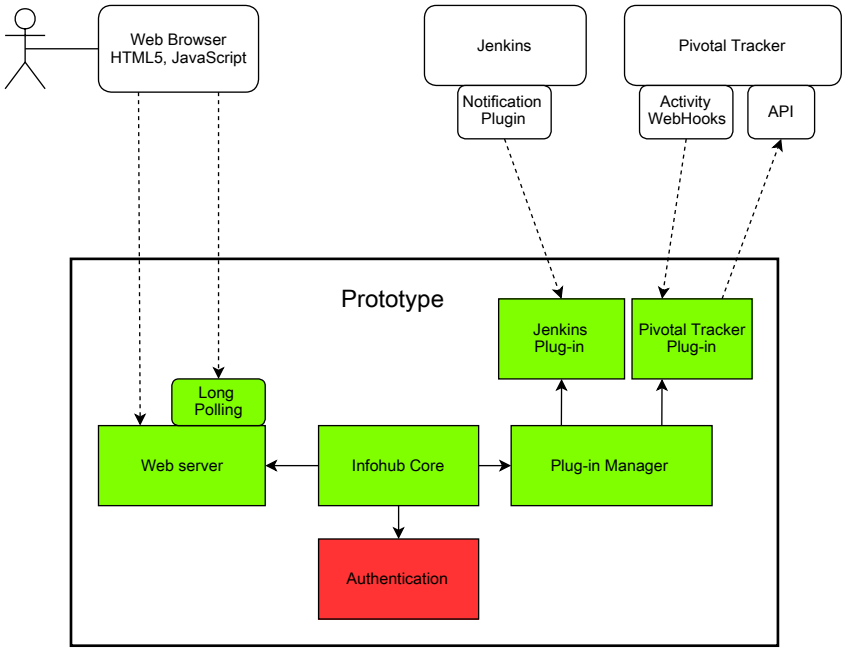

Figure 6: The modules that was implemented have a green background and the module that was not implemented have a red background.

Web Server

The prototype is written in the programming language Python, excluding the HTML5 web page and its JavaScript code. The web server part of the implementation uses the library Tornado. Tornado is a Python web framework that uses non-blocking network communication, which makes it ideal for long polling[10].

There is one web page that can be requested from the web server. This page contains the current information from all loaded plug-ins as well as a JavaScript function that is called automatically when the document is loaded. This function uses the JavaScript library jQuery[1] and is used to retrieve new information from the prototype and inserts it to the web page dynamically. The function uses the long polling method and the encoding JSON to accomplish this. The layout of the web page is dynamically created using the template system[2] in Tornado.

A screen shot of the web page that the prototype produces can be seen in Figure 7.

[1] See: `http://jquery.com/`
[2] See: `http://www.tornadoweb.org/en/stable/template.html`

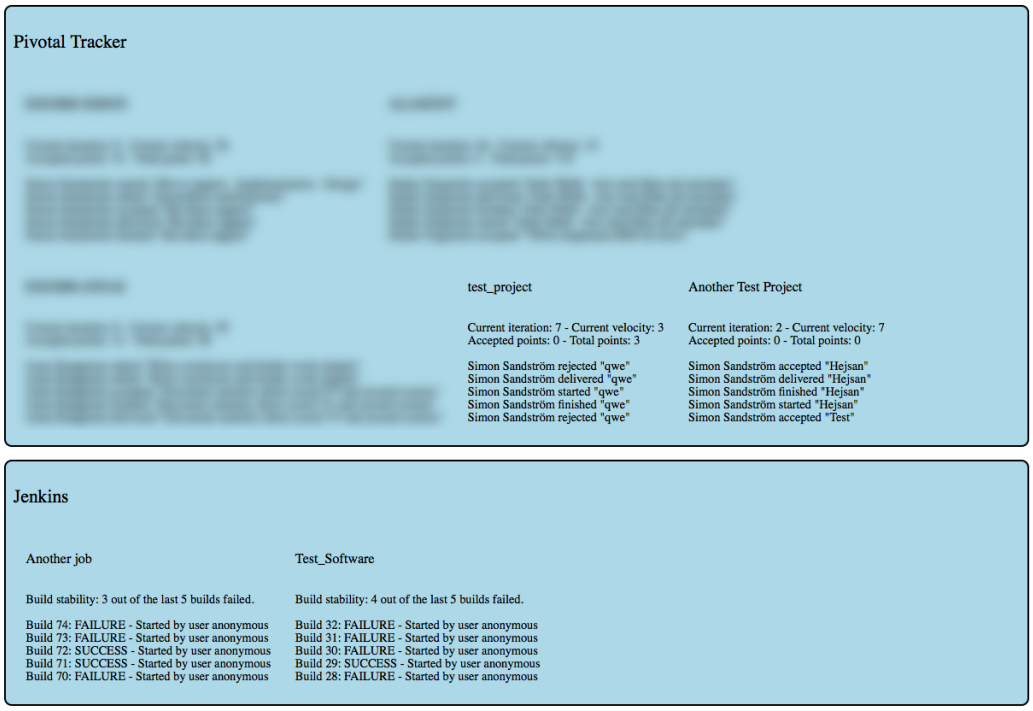

Figure 7: The web page that the prototype produces with the two plug-ins for Pivotal Tracker and Jenkins. The information about some projects from Pivotal Tracker are private and therefore blurred in this screen shot.

Plug-in Architecture

When the tool is started the plug-in manager searches a specific folder named *plugins* for all classes that derives from an abstract plug-in class. This class declares three methods which each plug-in can override to add functionality to the prototype. In addition to these methods, two class variables should be set by each plug-in. The class variables and methods are:

name A string with the name of the plug-in. This string can contain any type of characters and will be displayed on the web page.

internal_name A string with the internal name of the plug-in. This string can only contain lower case alphanumeric characters, which means the letters a to z as well as the numbers 0 to 9. The string can also contain underscores. This string is used for a number of things which is described later in this section.

get_content() This method should return the internal information of the plug-in as a string, which can include HTML. This method is required by all plug-ins.

poll() This method can be overridden if the plug-in wants to periodically do something, for example retrieve information from a remote server using its API. This method is optional to override for a plug-in and it should return *True* if new information now is available in the plug-in, otherwise *False*.

webhook(request) This method, if defined, is called when the web server gets an HTTP POST request to the path */plugins/[internal_name]/*, where *[internal_name]* is the

internal name of the plug-in as described previously in this subsection. The object *request* contains the included information of the POST request. As with the method *poll()*, this method is also optional and should return *True* if there is new information available and otherwise *False*.

Implemented Plug-ins

Two plug-ins was implemented for this prototype, one for retrieving project information from Pivotal Tracker and another for retrieving information about executed jobs from Jenkins. These two plug-ins will be described in detail in this subsection.

The Pivotal Tracker plug-in uses the activity webhook functionality in Pivotal Tracker as well as the Pivotal Tracker API to retrieve information. The plug-in is statically configured with a list of tuples. Each tuple contains a project id and a user token where the user token is used to gain access to the API as described in the Section 2.1. The information that this plug-in displays to the user of the prototype includes: the number of the current iteration, the current velocity of the project measured in points[3], how many points that are completed in the current iteration, how many points there are left to complete in the current iteration, the next milestone and when it is due and finally a list with the latest activity in the project.

When the plug-in is first initialized it accesses the Pivotal Tracker API and retrieves information about all the projects that are configured. The *webhook* method is defined so that the plug-in can be notified when there has been new activity in one of the projects. The plug-in actually does not use the information that is sent to the *webhook* method, instead it accesses the Pivotal Tracker API to update the internal information. For each project that is configured in the plug-in, one must enter the URL to the prototype in the Pivotal Tracker web interface so that it does send activity information to the prototype.

The Jenkins plug-in uses the same approach as the Pivotal Tracker plug-in. The *webhook* method is defined so that the plug-in is notified when a job have been executed on the Jenkins server. For each notification, the plug-in accesses the Jenkins API to retrieve status about the latest executions. The plug-in *Notification Plugin*, which is described in Section 2.2, needs to be installed on the Jenkins server and configured to send status of jobs that have been executed to the prototype web server.

This plug-in displays how many of the last five executions that have failed and a list of the the five last executions including if the execution succeeded or failed and what user started the execution.

User Interface

The user interface for the tool was designed with usability in mind. Usability is described in more detail later, in Chapter 5. Usability can be divided in to the five components; learnability, efficiency, memorability, errors and satisfaction. To make it easy to learn how to use the tool and how to remember to use the tool, it was designed to be as simple as possible. To avoid the possibility of a user making errors when using the tool, no buttons, options or links were used at all. Instead, all information from the plug-ins are displayed on a single web site. This also make the tool efficient to use.

[3]Described in Section 2.1

A user only need to enter the web site, and can then read all information that is available. As seen on Figure 7, the user interface is split in two parts. One part that contains information from the plug-in for Pivotal Tracker and one part with information from the plug-in for Jenkins. The information in each part is divided for each project. This makes it easy to find the correct information and to see from which tool this information belongs to.

4.3 Conclusion

As presented in Chapter 1, these are the requirements for a new tool:

- **R1**: It must be able to gather and display some information from atleast the two tools; Pivotal Tracker and Jenkins.
- **R2**: The tool must be free to use.
- **R3**: The tool must be web-based such as it can be accessed from a number of different devices and operating systems.
- **R4**: The tool must be designed in a way that makes it more efficient to use for the purpose of getting an overview of active projects than by using the different tools individually.
- **R5**: The tool must support authentication for different users.
- **R6**: It must be possible to choose what kind of information that should be displayed for each user.
- **R7**: The tool must support plug-ins for retrieving and displaying information from other tools. This means that atleast two plug-ins must be implemented due to requirement **R1**.

The requirements R1, R2, R3 and R7 are all met by the new tool. It does have plug-ins for both Pivotal Tracker and for Jenkins, it is free to use and it can be accessed with a web browser using the Internet. To determine if requirement R4 is met, one can not only look at the specifications of the tool, but to use some kind of usability test or evaluation. This is done and presented in Chapter 5. Requirement R5 and R6 are met by the system design of the new tool, but are not met by the actual implementation of the new tool.

5 Software Usability

There are two attributes of a software that together can measure how useful a software is. The first attribute is utility, which is if a user can use the software to complete certain tasks. The second attribute is usability, which estimates how easy a software or any other type of product is to use[11]. In this Chapter, usability is explained in detail. Different types of methods for evaluating usability is also presented.

Usability can be divided into these five components:

Learnability: Is it easy to learn how to use the product to accomplish certain tasks?

Efficiency: How fast can tasks be completed with the product?

Memorability: Is it easy for a user to remember how to use the product?

Errors: Does the user often encounter errors while using the product?

Satisfaction: Does the user feels satisfied when using the product?

It is very important that software is easy and efficient to use. Software that is developed and sold to customers should be designed with usability in mind, or else the customers might choose an alternative that is easier to use. Internal software that is developed and should be used by employees at a company should also be designed with usability in mind. Employees that have to spend too much time to accomplish some tasks lowers the productivity.

There are many methods for evaluating and improving usability in a software. These methods can be divided into two types, usability tests and usability evaluations[12]. These two types will now be described in detail. After these two types have been described, the usability evaluation method used to compare the reference procedure and the procedure that includes the use of a new tool will be described.

5.1 Usability Testing and Evaluation

Usability Testing

Usability testing is used to evaluate a software by testing it with representative users[13]. The purpose of performing usability tests is to identify problems with the software that must be corrected as well as collecting quantitative data. The quantitative data can show that some tasks takes too much time or that users often fail to accomplish certain tasks. Usability tests should be performed as early as possible so that problems can be corrected without the need to change major parts of the design or implementation.

There are at least two things that one must keep in mind when performing theses tests. The first is to test the software, not the users. The result of a usability test must only consists of

eventual flaws and problems with the software, not any problems with a user. The second thing to keep in mind is that the results of a usability test should be used to actually correct these problems.

The general method of a usability test can be described, slightly simplified, as the following steps:

1. Develop a test plan that specifies what to test, why to test it, when to test it and how to test it.
2. Create a list of tasks that the users should try to accomplish during the test.
3. Set up an environment where the test can take place.
4. Let the users, one by one, try to accomplish the given tasks. While the user is testing the software, the observer takes notes and other types of data. The data to collect is decided in the test plan.

Usability Evaluation

A usability evaluation is usually performed without any users. Surveys, cognitive walkthroughs, expert reviews and heuristic evaluations are some evaluation techniques. This type of evaluation is performed to find *possible* problems that can arise when a user uses the software. These problems can later be confirmed by performing a usability test with representative users, which was described in the previous section.

One of the most popular method is the heuristic evaluation[14]. In this method, the software is reviewed and compared against a set of heuristics. A heuristic is a ”rule of thumb,” or a good guide to follow when making decisions. The result of the evaluation is a list of potential usability issues. One of the best known developed heuristics are the one developed by Jakob Nielsen and Rolf Molich in 1990[15] and later refined by Jakob Nielsen in 1994[16]. These will not be explained in more detail in this report.

5.2 Comparison of the Different Procedures

To compare the reference procedure and a new procedure which uses the new tool, two different usability tests have been performed. The tests focuses on two types of users, project leaders and software developers. The reason for focusing on these to users is because a project leader is involved in a number of projects while a developer is typically only involved in one project. The tasks that have been performed in these tests are presented in Appendix A. The test was performed with two project leaders and two software developers. These steps describes how these tests was performed:

1. It is explained to the user why this test is being done and how the test is going to be performed.
2. The user reads the task that the user should try to complete and asks questions if anything is unclear.

3. Before the test starts the user should be logged out from Pivotal Tracker and Jenkins and not have any of those web pages open.
4. A timer is started and the user can begin the task.
5. The observer takes notes on how the user is completing the task. A list of quantitative and qualitative measurements that are collected during the tests is described later in this section.
6. The timer is stopped when the user thinks that the task is completed or if the user gets stuck and can not complete the task.

The measurements that these tests will collect focuses mostly on the efficiency aspect of usability, as the goal of the project is to find a way to make it faster and easier to get an overview of projects. Other aspects of usability, such as learnability and satisfaction, in the different procedures and their respective tools can be gathered based on the users comments for each test. Here follows a list of the quantitative and qualitative measurements that was collected with both of these tests:

- Comments by the user on each task.
- If the user successfully completed the task.
- The number of different software tools that was used in order to complete the task.
- The time to complete the task.
- The number of steps taken to complete the task. A step is defined as:
  - Clicking a button, clicking a link or similar to reach a new web page.
  - Entering any type of text, such as a URL or user credentials. Each text box with input from the user, both manually entered or automatically set by the web browser, counts as one step.

5.3 Conclusion

The raw results from these tests can be found in Appendix C. As only a few initial tests has been done no real conclusions can be drawn from the results. In order to get proper results and to be able to draw conclusions, more test and more complex tests must be done.

As stated in Chapter 4, to find if the new tool meets requirement R4, one must use usability tests or usability evaluation. Since these are inital and very simple tests, this can not be determined.

6 Discussion

The first goal of this project states: ”Find and evaluate any existing tools that might meet the requirements of a new tool.”. One existing tool was found and evaluated, as presented in Chapter 3, but did however not fit all of the requirements of a new tool. As the tool is open source it could have been modified to fit the requirements, but a decision was made that it would have taken longer time to do this than to develop a new tool.

The second goal of this project states: ”If no existing tool meets the requirements: design and implement a prototype of a new tool.”. Since the only tool found did not fit the requirements, a new tool had to be developed. The development of this tool was done in two different steps.

First the system design of the tool was developed. This design explains how the system should be divided and how it should work, internally. Many of the features in the system design was developed by looking at the requirements of a new tool, as well as the existing tool which was evaluated in Chapter 3. The system design does meet all of the requirements of a new tool.

The second step of this goal was to implement the system design to a working tool. The results of this step is a fully functional tool which however, only meets some of the requirements of a new tool.

The third goal of this project states: ”Evaluate the new tool in the aspect of usability to see whether it is a good solution and to find points of improvement.”. This goal have not been completely accomplished. Only some initial usability tests have been done. To be able to draw any conclusion about the new tool, and to tell whether this tool is efficient to use, more tests must be done.

6.1 Limitations

There are some limitations with the new procedure. The tool that is used in this procedure, which is presented in Chapter 4, can retrieve and display some information from the two tools Pivotal Tracker and Jenkins. But since this information must fit on a single web page, not all valuable information can be displayed. One way to deal with this problem was presented in the system design section of Chapter 4, namely to make it possible for specific users to log in to the system and configure what kind of information should be displayed to that specific user.

6.2 Future Work

As already stated in previous chapters, more usability tests and also even usability evaluations should be done. There are also many features that can be added to the tool that was developed in this project. The missing features that is described in the system design of the prototype should be implemented. As this gives the ability for every single user to customise the tool as he or she wishes, the procedure of getting an overview of projects using this tool can be even more efficient.

The requirements focused on the two tools Pivotal Tracker and Jenkins, but there are many more tools that are in use. So, plug-ins to the prototype that can retrieve and display information from these other tools and systems should be developed. There are many ideas of plug-ins that can be created and used. Here is a list of a few of them:

- Retrieve and display information from the company's internal schedule system, to be able to see if any employee is home sick, away on a business trip, away on vacation and so on.
- Retrieve and display information from social media that is of interest of the employees at the company.
- Retrieve and display sales information for products that the company have developed.

Glossary

This section contains a list with descriptions of terms that have been used through out this report, see Table 1.

Table 1: Description of terms used in this report

Term	Description
API	API is an abbreviation for Application Programming Interface. An API describes how software can communicate with each other.
Continuous Integration	Continuous Integration is a software development practice where members of a team integrate their work frequently to a shared code repository.
HTTP	HTTP is an abbreviation for Hypertext Transfer Protocol, which describes how hypertext should be transferred between hosts.
JavaScript	JavaScript is a programming language, originally implemented as part of web browsers.
JSON	JSON is an abbreviation for JavaScript Object Notation. It is a text based standard to encode data.
Open source	Open source generally refers to a software in which the source code is available to the general public for use and/or modification from its original design.
Polling	Polling refers to actively sampling the status of an external device by a client program as a synchronous activity.
Product backlog	In Scrum, each project have a prioritised list, called product backlog, of features that should be added to the product.
Scrum	Scrum is an iterative and incremental agile development method.
Sprint	In Scrum, each work cycle is called a sprint. Every sprint consists of clear goal that should be completed before the sprint ends. The time span for a sprint can range from one to a few weeks.
Sprint backlog	In Scrum, the sprint backlog is a prioritised list with the features that should be added to the product in the current sprint. The features in this list is gathered from the product backlog in the start of every new sprint.
Unit testing	Unit testing is a method for testing source code to find defects, flaws and other kind of bugs in software in development.
User Story	In Scrum, a feature that should be added to the product is often called an user story, or just a story.
Webhook	Webhook is a method for sending notification or other types of information from one system to another system when a certain event have occurred.

Continued on next page

Table 1 – *Continued from previous page*

Term	**Description**
XML	XML is an abbreviation for Extensible Markup Language, which is a language for encoding information in a format that is both human-readable and machine-readable.

Bibliography

[1] Pivotal Labs Inc. **Pivotal Tracker: Features**.
https://www.pivotaltracker.com/features.
Visited 07/05/2013.

[2] Jenkins CI. **Meet Jenkins**.
https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins.
Visited 07/05/2013.

[3] Jeff Sutherland Ken Schwaber. **The Scrum Guide**.
http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf.
Visited 10/05/2013.

[4] Pivotal Labs Inc. **Pivotal Tracker Help: API**.
https://www.pivotaltracker.com/help/api.
Visited 07/05/2013.

[5] Pivotal Labs Inc. **Pivotal Tracker Help: Integrations**.
https://www.pivotaltracker.com/help/integrations?version=v3.
Visited 07/05/2013.

[6] Sean Stolberg. **Enabling Agile Testing Through Continuous Integration**.
http://www.agilemethod.csie.ncu.edu.tw/agilemethod/download/2009papers/2009%20Enabling%20Agile%20Testing%20Through%20Continuous%20Integration/Enabling%20Agile%20Testing%20Through%20Continuous%20Integration.pdf.
Read 07/05/2013.

[7] Pivotal Labs Inc. **ProjectMonitor offical web page**.
https://github.com/pivotal/projectmonitor.
Visited 07/05/2013.

[8] Djane Rey Mabelin. **Long Polling vs Short Polling**.
http://codertalks.com/long-polling-vs-short-polling/.
Visited 20/05/2013.

[9] Dorian Birsan. **On Plug-ins and Extensible Architectures**.
http://queue.acm.org/detail.cfm?id=1053345.
Visited 20/05/2013.

[10] Tornado Official Web page. **Main web page**.
http://www.tornadoweb.org/en/stable/.
Visited 21/05/2013.

[11] Jakob Nielsen. **Usability 101: Introduction to Usability**.
http://www.nngroup.com/articles/usability-101-introduction-to-usability/.

Visited 10/05/2013.

[12] U.S. Department of Health & Human Services. **Usability.gov: Types of Evaluations**.
http://www.usability.gov/methods/test_refine/learneval.html.
Visited 10/05/2013.

[13] U.S. Department of Health & Human Services. **Usability.gov: Usability Testing**.
http://www.usability.gov/methods/test_refine/learnusa/index.html.
Visited 10/05/2013.

[14] U.S. Department of Health & Human Services. **Usability.gov: Heuristic Evaluations**.
http://www.usability.gov/methods/test_refine/heuristic.html.
Visited 10/05/2013.

[15] Jakob Nielsen Rolf Molich. **Improving a Human-Computer Dialogue**.
http://ouvea.edu.ups-tlse.fr/~truillet/ens/m2iaici/articles/p338-molich.pdf.
Visited 26/05/2013.

[16] Jakob Nielsen. **Enhancing the Explanatory Power of Usability Heuristics**.
http://www.cs.helsinki.fi/u/salaakso/kl2-2002/lahteet/Nielsen94-Enhancing-Heuristics.pdf.
Visited 26/05/2013.

A Usability Test Tasks

This appendix presents the task that was given to project leaders and software developers to accomplish in the usability tests.

Task for project leaders

The task for project leaders consists of these three steps:

1. For each of the projects that you are leading, find how much work that have been completed this week and how much work that should have been completed at the end of this week.
2. For each of the projects that you are leading, find what and when the next milestone is and check whether the milestone will be reached in time.
3. For each job in Jenkins which belongs to a project that you are leading, find the status of the latest job execution and who started the execution.

Task for software developers

The task for software developers consists of these three steps:

1. Find how much work that have been completed this week and how much work that should have been completed at the end of this week of the project that you are currently working with.
2. Find what and when the next milestone is and check whether the milestone will be reached in time of the project that you are currently working with.
3. If the project that you are currently working with have one or more jobs in Jenkins that belongs to the project, find the status of the latest job executions of these jobs.

B Technical Details Examples

This appendix presents a few examples of technical details from the two tools Pivotal Tracker and Jenkins, which is used in the reference method.

Pivotal Tracker: API examples

These examples is taken directly from the official help manual for Pivotal Tracker API version 3, which can be found at `https://www.pivotaltracker.com/help/api?version=v3`. The examples uses the tool *curl* to make HTTP requests to the API.

Retrieving the recent activity of a specific project with id *project_id* with a user token *user_token_x*:

```
Executed command:
curl -H "X-TrackerToken: user_token_x" -X GET \\
    "http://www.pivotaltracker.com/services/v3/projects/project_y/activities?limit=50"

Output:
<?xml version="1.0" encoding="UTF-8"?>
<activities type="array">
    <activity>
        <id type="integer">1031</id>
        <version type="integer">175</version>
        <event_type>story_update</event_type>
        <occurred_at type="datetime">2009/12/14 14:12:09 PST</occurred_at>
        <author>James Kirk</author>
        <project_id type="integer">project_id</project_id>
        <description>James Kirk accepted &quot;More power to shields&quot;</description>
        <stories type="array">
        <story>
            <id type="integer">109</id>
            <project_id type="integer">project_id</project_id>
            <url>https://www.pivotaltracker.com/services/v3/projects/project_id/stories/109</url>
            <accepted_at type="datetime">2009/12/14 22:12:09 UTC</accepted_at>
            <current_state>accepted</current_state>
          </story>
        </stories>
    </activity>
</activities>
```

Pivotal Tracker: Activity Webhooks examples

This example was done by creating a project in Pivotal Tracker and configure the Activity Web Hook to send project activity information to a service called *RequestBin*[1]. RequestBin is used to inspect incoming HTTP requests. This is the response after creating a user story with the type *Feature*, title *Dummy Feature* using the web interface for Pivotal Tracker:

```
POST /oh124roh HTTP/1.1
Host: requestb.in
Content-Type: application/xml
Content-Length: 968
Connection: close
Accept: */*

<?xml version="1.0" encoding="UTF-8"?>
<activity>
  <id type="integer">359133137</id>
  <version type="integer">81</version>
  <event_type>story_create</event_type>
  <occurred_at type="datetime">2013/05/15 14:06:17 UTC</occurred_at>
  <author>Simon Sandstr&#246;m</author>
  <project_id type="integer">797067</project_id>
  <description>Simon Sandstr&#246;m added &quot;Dummy feature&quot;</description>
  <stories type="array">
    <story>
      <id type="integer">49925149</id>
      <url>http://www.pivotaltracker.com/services/v3/projects/797067/stories/49925149</url>
      <name>Dummy feature</name>
      <story_type>feature</story_type>
      <description>Just a dummy feature for testing the Activity WebHook.</description>
      <estimate type="integer">2</estimate>
      <current_state>unscheduled</current_state>
      <owned_by>Simon Sandstr&#246;m</owned_by>
      <requested_by>Simon Sandstr&#246;m</requested_by>
    </story>
  </stories>
</activity>
```

[1] See: http://requestb.in/

Jenkins: Notification Plugin examples

These examples was done by doing a clean install of Jenkins version 1.424.6 and adding a job named *Test*. The job was configured to execute a shell script that returned an exit status of 1 to simulate a script that failed. After confirming that the job could be executed using the web interface of Jenkins, the plug-in *Notification Plugin* was installed and configured to send execution information to a service called *RequestBin*[2]. RequestBin is used to inspect incoming HTTP requests.

This is the information sent to RequestBin when starting the execution of the created job:

```
POST /12engqi1 HTTP/1.1
User-Agent: Java/1.6.0_27
Host: requestb.in
Content-Type: application/x-www-form-urlencoded
Content-Length: 119
Connection: close
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2

{
    "name":"Test",
    "url":"job/Test/",
    "build": {
        "number":9,
        "phase":"STARTED",
        "url":"job/Test/9/"
    }
}
```

This is the information sent to RequestBin when the execution of the created job finished:

```
POST /12engqi1 HTTP/1.1
User-Agent: Java/1.6.0_27
Host: requestb.in
Content-Type: application/x-www-form-urlencoded
Content-Length: 139
Connection: close
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2

{
    "name":"Test",
    "url":"job/Test/",
    "build": {
        "number":9,
        "phase":"FINISHED",
        "status":"FAILURE",
        "url":"job/Test/9/"
    }
}
```

[2]See: http://requestb.in/

C Result of Usability Tests

In this Appendix, the result data of the usability tests that have been conducted is presented. The result is divided and presented per user. For each user, first the result of the usability test of the reference procedure is presented, then the result of the usability test of the new procedure.

Result of the Usability Test with Software Developer 1

Reference Procedure
User comments: None. **Completed the task**: Yes. **Steps taken**: Enters the Pivotal Tracker web page. Clicks the Sign In-link. Enters user name. Enters password. Presses the login button. Selects the project. Enters the Jenkins web page. Selects the project. Selects the latest execution. **Tools used**: Pivotal Tracker, Jenkins. **Time taken**: 110 seconds.
New Procedure
User comments: None **Completed the task**: Partially: could not see if the next milestone was predicted to be completed before the deadline, the user had to calculate it himself. **Steps taken**: Enters prototype web page. **Tools used**: Prototype. **Time taken**: 36 seconds.

Result of the Usability Test with Software Developer 2

Reference Procedure
User comments: None. **Completed the task**: Yes. **Steps taken**: Enters the Pivotal Tracker web page. Clicks the Sign In-link. Enters user name. Enters password. Presses the login button. Selects the project. Enters the Jenkins web page. Selects the project. Selects the latest execution. **Tools used**: Pivotal Tracker, Jenkins. **Time taken**: 90 seconds.
New Procedure
User comments: None **Completed the task**: Partially: could not see if the next milestone was predicted to be completed before the deadline, the user did not try to calculate it himself. **Steps taken**: Enters prototype web page. **Tools used**: Prototype. **Time taken**: 59 seconds.

Result of the Usability Test with Project Leader 1

Reference Procedure
User comments: ”I would like to get an overview of specific project members, how much work they have completed and how much work that they have planned to do this iteration. Currently it is only possible to see this information for each project, which means that I need to summarise the work for each project member manually.” **Completed the task**: Yes. **Steps taken**: Enters the Pivotal Tracker web page. Clicks the Sign In-link. Enters user name. Enters password. Presses the login button. Selects *Project A*. Selects *Project B*. Selects *Project C*. Enters the Jenkins web page. Selects *Project A*. Selects the latest build. Selects *Project B*. Selects the latest build. Selects *Project C*. Selects the latest build. **Tools used**: Pivotal Tracker, Jenkins. **Time taken**: 472 seconds.
New Procedure
User comments: ”I would like to see different colors based on if a project’s job have executed successfully or with any error in Jenkins, and also the specific time of an execution.” **Completed the task**: Partially: could not see if the next milestones for each project was predicted to be completed before the deadline, the user had to calculate it himself. **Steps taken**: Enters prototype web page. **Tools used**: Prototype. **Time taken**: 97 seconds.

Result of the Usability Test with Project Leader 2

Reference Procedure
User comments: None. **Completed the task**: Yes. **Steps taken**: Enters the Pivotal Tracker web page. Clicks the Sign In-link. Enters user name. Enters password. Presses the login button. Selects *Project A*. Selects *Project B*. Selects *Project C*. Enters the Jenkins web page. Selects the build history link. **Tools used**: Pivotal Tracker, Jenkins. **Time taken**: 239 seconds.
New Procedure
User comments: None. **Completed the task**: Partially: could not see if the next milestones for each project was predicted to be completed before the deadline, the user had to calculate it himself. **Steps taken**: Enters prototype web page. **Tools used**: Prototype. **Time taken**: 156 seconds.