How can Atlassian products be modified to reduce the average time usage for common tasks

Anthon Johansson

Supervisor : Anders Fröberg
Examiner : Erik Berglund
Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innefattar rätt att bli nämnt som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida [http://www.ep.liu.se/]

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page [http://www.ep.liu.se/]

© Anthon Johansson
Abstract

Software tools such as Build systems and project management tools are sometimes not well designed when it comes to usability. This paper investigates the possibility of creating custom solutions for the three Atlassian products Jira, Confluence, and BitBucket, in order to increase the efficiency when performing common tasks at the Atlassian administration at Saab AB. It was discovered that the main issue was performing reoccurring project-access permission audits, which was a very repetitive task with many manual steps involved. The solution to the issue was a Python script that, through the use of the Atlassian REST API, could collect all the necessary information automatically and present it in a readable summarized view. The amount of manual steps was significantly decreased to just a few steps which made a huge difference for the Atlassian administrators at Saab AB.
Acknowledgments
### Contents

- **Abstract** iii
- **Acknowledgments** iv
- **Contents** v
- **List of Figures** 1

#### 1 Introduction
- 1.1 Motivation .............................................. 3
- 1.2 Aim ....................................................... 3
- 1.3 Research questions .................................... 3
- 1.4 Delimitations ........................................... 3

#### 2 Atlassian
- 2.1 Atlassian ............................................... 4
- 2.2 Atlassian REST API ................................... 5
- 2.3 Atlassian Software Development Kit ................. 5
- 2.4 REST API vs SDK ....................................... 5

#### 3 Theory
- 3.1 Build System ........................................... 7
- 3.2 Continuous Integration ................................ 7
- 3.3 Continuous integration based on pull-requests .......... 10
- 3.4 Web Service .......................................... 11
- 3.5 REST API ................................................. 12
- 3.6 Degree of Automation .................................. 12
- 3.7 Benchmark test ......................................... 13

#### 4 Method
- 4.1 Pre-study ............................................... 14
- 4.2 Deciding the benchmark test tasks ....................... 14
- 4.3 Planning of iteration ................................... 15
- 4.4 Development work ..................................... 15
- 4.5 Evaluation and reflection ................................ 15

#### 5 Results
- 5.1 Pre-study ............................................... 17
- 5.2 Benchmark test ........................................ 19
- 5.3 Iteration 1 ............................................. 19
- 5.4 Iteration 2 ............................................. 22

#### 6 Discussion

v
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>An example of the JSON output from the Jira REST API.</td>
<td>25</td>
</tr>
<tr>
<td>5.2</td>
<td>An example of the role specific JSON output from the Jira REST API.</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>An example of the JSON output from the BitBucket REST API.</td>
<td>26</td>
</tr>
<tr>
<td>5.4</td>
<td>An example of the JSON output from the BitBucket REST sub-resources</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>&quot;/{projectkey}/permissions/groups&quot; and &quot;/{projectkey}/permissions/users&quot;.</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>An example of the JSON output from the BitBucket REST sub-resources</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>&quot;/{projectkey}/repos/{repositorykey}/permissions/group&quot; and &quot;/{projectkey}/repos/{repositorykey}/permissions/users&quot;.</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>An example of the text file containing the summary of Jira project permissions.</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>&quot;TEST&quot; &amp; &quot;TEST2&quot; is the projects and &quot;group&quot; &amp; &quot;user&quot; is the user groups or users</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that has been added to each role in each project. Each role has different permission settings.</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>The tool menu.</td>
<td>28</td>
</tr>
<tr>
<td>5.8</td>
<td>An example of how the Jira diff file works, the info marked green has been added</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>and the info marked red has been removed from the latest permission settings as compared to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the chosen log entry 22/05/2017.</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>An example of how the BitBucket comparison file could look like.</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>&quot;TEST2&quot; is the project key and &quot;group&quot; &amp; &quot;user&quot; contains all the user groups or users that has</td>
<td></td>
</tr>
<tr>
<td></td>
<td>been added to a specific permission. Other details such as if the project/repository is public</td>
<td></td>
</tr>
<tr>
<td></td>
<td>is visible too.</td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>An example of how the generated BitBucket diff file could look like. Figure 5.8 gives a</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>detailed explanation of how it works in the Jira case.</td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction

The business area of software development has been around for many decades now and has been used in all kinds of projects, from large complex projects such as space missions to small scale projects such as simple scripts for websites. What both of these types of projects have in common is not only that programming is involved but also that management systems are used. When the workforce and work tasks goes from one programmer working on one task to many programmers working on many tasks in many simultaneous projects, management will be a crucial activity.

However, while these management systems improve the organization they also introduce new problems, namely problems related to Human-Computer-Interaction (HCI). Many software products are fully functional for the purpose they were designed for, but the human factor greatly affects the efficiency when used. Simple solutions are often the best solutions and the same principle applies to this situation. A system that is (and looks) easy to use will increase the perceived ease of use of the intended user, familiarity with the system will increase it even further. This implies that small changes to an existing system are easier to adapt to rather than adapting to an entirely new system. Studies also show that anxiety doesn’t have a big effect on perceived ease of use. [15]

Atlassian

Atlassian is a software company that was founded the year 2002 in Australia by Scott Farquhar and Mike Cannon-Brookes. Atlassian offers software products for activities such as project management and software development.[5] The software products offered by Atlassian are used by many companies in the world and new software products are added to the list of software products as time goes on, for example Bitbucket and Trello. [2][3]

The owners of Atlassian and all its software products have also realized that there are a need of customization to their software products. Company A might need something entirely different than company B, every Atlassian user might have its own customization requirements. Satisfying the need of every Atlassian user would require a lot of resources, so
in order to solve this issue they released a REST API\textsuperscript{22} and a plugin software-development-kit (SDK) for developing custom add-ons to each Atlassian product.\textsuperscript{8}

1.1 Motivation

As mentioned above, the addition of new software products also includes training time for the intended working staff and in some situations a user interface design that is anything but friendly to human eyes. It is common for people working with computers to have computerphobic thoughts when operating the software products.\textsuperscript{15} In some cases it is just a matter of computer habit/familiarity with the system and will gradually disappear after some time of use. However, there are also situations where the software product could use some customization on the visual and functional level to increase the work efficiency and software usability.

1.2 Aim

The goal of this thesis project is to, based on interview data and user tests, improve the functionality of the Atlassian products used by the Atlassian administration at the company Saab Group AB in Linköping, Sweden. Preferably by making many of the common tasks automated to some degree to reduce the amount of human errors when operating the Atlassian products. Some of the tools used at the company are not part of any Enterprise System, such as scripts and third party programs.

1.3 Research questions

To reach this goal the following research question will be answered.

1. How should the customization to the Atlassian products be designed to achieve minimal time usage for common tasks?

1.4 Delimitations

Since this project will be using the Atlassian plugin SDK\textsuperscript{1} and REST\textsuperscript{2} API\textsuperscript{3} for the customization process, what they have to offer is a limitation. Since this project is similar to a user-experience design project, it also has the time limitation. Designing something can go on forever in several iterations, it is ultimately the deadline of the project that decides the stopping point.

\textsuperscript{1}Software Development Kit
\textsuperscript{2}Representational State Transfer
\textsuperscript{3}A set of routines, protocols and tools for building software applications.
This chapter brings up some background info about Atlassian and some of the software products that they have to offer.

2.1 Atlassian

Atlassian was founded in the year 2002 in Australia by Scott Farquar and Mike Cannon-Brookes. The company has grown a lot since then and now offer several software services to customers. Atlassian has a range of products to offer which can be used in several different usage areas. The products used by Saab are listed below with a short description.

Jira

Jira is a product used for planning, tracking and support purposes. In this case Saab is only using Jira Software, which is used for project planning and issue tracking. There are also other versions of Jira such as Jira Service Desc and Jira Core. This product allows the user to add features such as elements from the Scrum or Kanban methodology. It also offer the option to use long term planning features, to make long term planning easier by offering more visibility over all the projects and teams. There are a number of additional tools and elements that can be used in Jira Software. Developer tools can be integrated if needed, such as Bitbucket or GitHub, and add-ons can be installed to enable custom functionality.

Confluence

Confluence works much like other popular tools such as Microsoft SharePoint or Google Drive, it is a central point for sharing knowledge with every member of the team or organization. This allows fast synchronization between workers, customers or any other person who may need access to the information. Special configurations and other customizations can be
2.2 Atlassian REST API

The Atlassian REST API is a collection of restful web services that can be used to remotely access the services on the Atlassian platform. Information such as Jira issue details, project members, permission settings etc can be extracted through the REST API. It is also possible to change settings or modify information in any of the Atlassian products through the REST API.[14]:

2.3 Atlassian Software Development Kit

The Atlassian Software Development Kit (SDK) allows developers to create plugins that will extend the functionality of the Atlassian applications. There are several common components (with APIs) that exist to make the development process as smooth as possible. The plugins are written in the programming language Java and can call methods in each of the APIs in the common components.

2.4 REST API vs SDK

There are some major differences between these two ways of customizing your Atlassian application usage experience.

The REST API is very universal since the communication is done through HTTP method calls (get, put, post and delete) where the service response is delivered in text format (XML, JSON etc.). This means that you can use the Atlassian platform services in any web browser, or in third party clients written in any programming language that supports HTTP method calls. There are often a lot of available web services out there where the functionality in each service is versatile.

Unlike the REST API, the plugin will only work with the existing Atlassian applications

---

1Hyper Text Transfer Protocol
2.4. REST API vs SDK

and not any external or third party client. So the user is still bound to the Atlassian applications and their current flaws and restrictions. The whole purpose of the plugin is to work around these restrictions but they are still affected by the existing components in the SDK, which is a restriction. On the other hand, these plugins are being run locally as an extension of the Atlassian application, so they do not depend on an internet connection to function.
This chapter brings up related research and theory relevant to the development and evaluation of the customization process in the project.

3.1 Build System

A general description of a build system is a system that transforms one form of data into another form of data. It is a central part of a software development project since normal work tasks such as testing, code changes, version control, code builds etc. are performed in the build system and many project members will be in contact with it. But these build systems are not always functioning properly and may cause issues, issues that end up in maintenance of the project builds. Some examples of this are cryptic error messages and new patches that create bugs in the build system, just daily struggles developers have to deal with on regular working hours. An empirical study of 10 long-lived software projects showed that, due to minor flaws in the build system, various steps in the development chain may be affected by overhead. [10] The goal of the build system is to create all the deliverables of the project and the executable package that can be used by the intended customer.

3.2 Continuous Integration

In many professions where the workforce is big there are sometimes communication errors that result in production issues, which will cause production delays. A product is often put together by several different teams and without proper communication these teams will most likely (at some point) make a mistake that affects the progress of other teams. Management work is important because of this reason, but there are some things that not even a manager will be aware of. Software development delays are often caused by undiscovered bugs or errors that somehow slips by many developers for a long time. These undiscovered errors

\footnote{An executable artifact created from source code}
and bugs will in the end affect the work progress in the shape of additional maintenance work which could have been avoided. The usual reason for this evasion of detection is in many cases due to the fact that new source code isn’t integrated to the main repository often enough.

Continuous integration is a set of best practices designed with the purpose of easing the integration process and speeding up the production pace in software development projects. Doing frequent code integrations will save time and effort in the long run, and will detect errors early on in the development process to minimize the amount of maintenance work. Some of these practices are performed automatically while others have to be performed manually. Some of these best practices are:\[4\]:

1. Maintain a code repository
2. Automate the build
3. Self testing build
4. Frequent daily commits to baseline branch
5. Build every commit to baseline branch
6. Rapid build speed
7. Test in the correct environment
8. Availability of latest deliverables
9. Visible build results and log
10. Continuous deployment

**Maintain a code repository**

All the source code should always be placed in a central code repository, the included version control system tools allows developers to easily maintain the integrity of the source code. There are several branching strategies in use for code repositories, one of the more popular one is to make new branches for new functions or code changes. The main branch in the repository should always contain the most recent functional version of the source code. New branches are made for every new code modification and the original branch content is always a copy of the (at the time) most recent version of the content in the main branch. Whenever a new functionality is finished and committed to the main branch it will go through unit testing and integration testing.

**Automate the build**

Systems such as Make has been around for several years and allows developers to automate the build process of source code. The build steps are written in a file which are read by the builder as it creates executable files from source code. This kind of build method reduces the chance of human errors occurring during the build process. Most of the build systems being used by software developers often has this as a built in feature.
3.2. Continuous Integration

**Self testing build**

Even if the source code actually builds into deliverables, without any build errors occurring during the process, there is still no guarantee that the deliverables functions properly. Some compilers are very strict (such as the Ada compiler) and will give compilation errors on the slightest breach of syntax rules, while other compilers are less strict. Common problems are memory leaks or application functions that cause run time errors. Both of these problems occur during run time and therefore should be covered by run time tests. These tests are often covered by automatic unit tests and automated integration tests on a dedicated build server. A team of dedicated testers will in almost any case sit and test every function manually after a build has passed the automatic tests and been deployed to a live test server.

**Frequent daily commits to baseline branch**

As mentioned earlier, new functions are always started with a fresh copy of the latest project files in the main branch. This is done by every developer and occur at random times and days depending on when the developers start working on a new function. The issues will start occurring when the main branch is updated with a new feature, one that isn’t in the developers temporary working branch. The version control system is built to handle synchronization problems like this but when the differences in the branches are too big it becomes extra troublesome and time-consuming to sort out. That is why every developer should commit their new completed features to the main branch as soon as possible.

**Build every commit to baseline branch**

As previously mentioned, the source code should always go through a build test to see if all the system components compiles properly. This should be done as soon as new source code is committed to the main branch and will be followed by automatic unit tests and integration tests.

**Rapid build speed**

Build speed is an important factor that affects productivity, spending a long duration waiting for a build to finish is very time-consuming. Slow build speeds is not a huge problem if the build process starts over night, but that also limits the developers to one build attempt each day. One solution to this is a central dedicated build server, which takes care of compilation, unit tests, and integration tests. Short build times is crucial since developers need to know about errors as soon as possible.

**Test in the correct environment**

Test environments are often used when tests are carried out and depending on the similarity between this environment and the production environment used by the customer, the tests might give different results. It can be a small variation such as the version number of a third party program or physical specifications of hardware. Virtual machines hosted by cloud companies are very useful in these cases and can virtualize the hardware being used by the customer, it is also scaleable which is useful in test scenarios.
3.3. Continuous integration based on pull-requests

**Availability of latest deliverables**

As previously mentioned, different developers will most likely base a new functionality on source code of different versions in the main branch. When a new function is finished it will have to go through a merging process in order to properly merge with the main branch. This merging process has to pass without affecting the integrity of the source code or creating new bugs. However, if the project files used by the developer are somewhat outdated there is a big risk that many merging conflicts will appear and affect the integrity of the source code, possibly creating new bugs or errors. To prevent this, developers should update their project files at least once a day and resolve the merging conflicts early on when they are small and less complex. As a result, testers will test the most recent build with the correct functionality and less errors or bugs.

**Visible build results and log**

To avoid keeping unnoticed bugs or errors in the source code, there should be a log with build results to keep every developer updated. It should display who the committing developer is and which changes that caused the build to break.

**Continuous deployment**

Most of the activities in continuous integration concerns automatic integration, unit testing, and integration tests of source code changes committed to the main repository. This can be taken one step further by automating the deployment of the build to a live test server environment. The acceptance tests, which usually are done manually by testers, will be performed automatically as well. These additional activities are part of the continuous delivery process. The final step is to automate the deployment of the build to a live production server environment, this is known as continuous deployment.

Continuous deployment (and the prerequisites from continuous integration/-delivery) can, if done properly, increase the productivity of the development work. It is a great approach for web-based technology such as Google cloud services where everything is done in a web browser. Most of the effort can be put into development and users will receive the latest build version by refreshing the web page in the browser. It is a less suitable solution for companies with strict release schedules of their products, i.e. companies who deliver a big update to their product line once a year.

3.3 Continuous integration based on pull-requests

A pull-request is a function that exists on source code hosting services such as Github, where most of the software projects are open to the public. Every repository on Github can be “forked”, which means that a copy of the repository is made on a different Github account. The pull-request will be made whenever a user wants to merge all the changes made in a forked repository into its original repository.

The interesting part of all this is that some of the available tools on Github allows the combination of forks and pull-requests to function as a continuous integration system. Travis-CI [17], Jenkins[12] and similar tools functions as a continuous integration and delivery system for source code repositories. Every pull-request will go through an automatic build test fol-
lowed by automatic integration and unit tests. If the tests pass these automatic filters the pull-request will be reviewed by one of the core developers of the original repository. The core developer will decide if the quality and function of the code modification are suitable to be merged into the repository. A study of several Github projects using this approach showed that the use of continuous integration on Github repositories increased productivity without affecting the code quality negatively.\textsuperscript{13} As a result of using the system, there was a big increase in reported (and fixed) bugs, since the increase in processed pull-requests (due to less manual administration work) encourages more developers to contribute more frequently.

3.4 Web Service

Web services have been in use for over a decade now and is among the available online web content along with obvious everyday web content such as pictures, videos, music etc.\textsuperscript{9} The word service does in this context translate to function, i.e. planning a flight route or converting between currencies. Instead of having this kind of functionality available through locally installed software it is available online, either through a license or as a free web service. It is also possible to offer the service within a internal network.

There are two main categories of web services, these are traditional web services and restful web services. Both of these categories of web services provides services online (or through networks) but they do it in different ways which affects their applicability in different situations.

SOAP based Web Services

Traditional web services are driven by SOAP\textsuperscript{2} messages\textsuperscript{[p72-77, 6]} and offer a set of operations described in a WSDL\textsuperscript{3} file written in XML\textsuperscript{4} format.\textsuperscript{[p67-72, 6]} The point of the WSDL (version 2.0) file is to describe the interface of a web service, with content such as:

1. How to connect to and call the web service
2. Which input parameters does the web service require, which ones are optional?
3. How the requesting client should process the SOAP message returning from the web service

It does work but all of these messages and formats are often confusing to human eyes. The WSDL will describe how a XML formatted message should be written in order to use the service, but these messages are written manually and can become quite complex.\textsuperscript{16} A machine can easily read the info in the WSDL file but it has a way of confusing human users who are interpreting it. However, the written XML message allows the web service to notify the user about errors in the XML message and how to fix them.

RESTful Web Services

RESTful web services was created as a second generation web service to the older traditional web service. The biggest change was the change of simplicity in usage, an area where SOAP

\textsuperscript{2}Simple Object Access Protocol.
\textsuperscript{3}Web Service Description Language.
\textsuperscript{4}Extensible Markup Language.
messages have failed. Instead of relying on and interpreting a unique WSDL file with unique methods, this kind of web service uses a uniform interface. This is possible due to the usage of the standard HTTP methods Get, Put, Post, and Delete which also allows simple re-use of the service. Many other aspects such as performance, scalability and reliability were enhanced as well. The returning message from the service will arrive in a markup language format, such as XML or JSON\(^5\). While this kind of web service is easier to use, it does not handle errors as efficiently as the traditional web service.

3.5 REST API

A REST API is just like any other API, it is a structured communication interface used for programs that are communicating with one another. If program X wants to use program Y, it has to do it in a specific way and will receive the response/output in a specific way, all of this is dictated by the API. In this case the API will describe how users or other programs can use a specific restful web service, much like the WSDL file in a traditional SOAP based web service. Just like WSDL files can offer several operations, a REST API can contain several restful web services.\(^2\)

3.6 Degree of Automation

The Degree of Automation (DoFA), is a model where the nature and amount of tasks to be executed by an operator are taken into consideration when measuring how autonomous a system actually is. Some tasks are often very simple and system performance may increase if an automatic system would perform the task. Other tasks may require the attention of an operator and would decrease the system performance if it were to be performed automatically by a system. Many researchers have tried to categorize tasks in various ways but DoFA uses the following rather simple categories:

1. Tasks which can only be performed by an operator
2. Tasks which can be performed by an operator or an automated system
3. Tasks which can only be performed by an automated system

Only tasks of category one and two are used in the model since DoFA is intended for measuring how automatic execution of a previously manual task might affect the system performance.

After the model was tested in experiments by the model creators it was concluded that having a partly autonomous system was about as efficient as a fully autonomous system, depending on where the limit of task automation was set. As the DoFA increases, the system performance becomes less and less sensitive to changes in DoFA. This is logical since it is mostly simple repetitive tasks that is time-consuming when performed manually. More complex tasks that require logical thinking is often equally time-consuming when performed by an operator. The connection between DoFA change and system performance suggests that having an operator would cause a trivial degrade in the system performance of a partially automatic system, once the DoFA is high enough.\(^{19}\)

\(^5\)JavaScript Object Notation
3.7 Benchmark test

A benchmark test is a universal way of testing the performance of something and exists in many different shapes. Benchmark tests serve as a good comparison mark simply because a series of standard tests are performed, which means that the same assessment criteria applies to every unit that goes through the test. The benchmark in this project consists of a collection of operator tasks, which are made up of a series of manual operator instructions.
This chapter will describe how the work will be carried out in the thesis project.

4.1 Pre-study

The Atlassian REST API and plugin SDK are the key to modifying the working procedure of the Atlassian products, so knowledge about both of these need to be acquired before any changes can be made. This was acquired by practical testing of both technologies where small example projects were created. Insight about how the Atlassian administrators at Saab actually use the Atlassian products will be very useful as well, this was acquired through interview questions.

Before any improvements can be made there needs to be a session where each Atlassian administrator can describe the current problems that they experience when they use the Atlassian products. This initial interview will be the start of each development iteration and will be a crucial part of the modification process. The first interview before iteration one will reveal most of the problems and the future interviews before future iterations will reveal the problems that was left after or was caused by the previous iteration. The interview will not be led in any particular direction, it is better if the Atlassian administrators themselves get to describe the problems that they consider to be obstacles in their usual work tasks.

4.2 Deciding the benchmark test tasks

Statistics such as the amount of used GUI clicks of the operator will be recorded in the test. The test tasks will be decided before any development work begins and will be chosen in a way that ensures that the usual daily working tasks, performed by the Atlassian administration, are examined. The initial benchmark test will serve as a starting point and each iteration will aim to improve these benchmark values.
4.3 Planning of iteration

Before the development work in the iteration starts there will be a planning session to schedule all the work in the iteration. This is to ensure that the iteration follows a set plan and stays within the time limit of the iteration. Each problem described in the initial interview will be analyzed and a general development plan will be decided for each problem. This analysis of each individual problem will cover the following questions:

1. Is it possible to solve the problem at all?
2. Should the problem be divided into smaller sub-problems?
3. Is the solution worth the time and effort that will be put into fixing the problem?
4. Is the REST API or the plugin SDK best suited for solving the problem?
5. What is the general design of the solution?
6. When is the problem considered fixed, which requirements must be fulfilled by the solution?

All of these analysis points will be used when creating the development plan for each problem solution. If the tasks in the current iteration is important and needs additional time the iteration might be extended by a few days.

4.4 Development work

The development work will be performed in a suitable integrated development environment (IDE) such as Eclipse or similar that supports the programming languages Java and Python. The solutions will make use of either the Atlassian plugin SDK or the REST API, this depends entirely on the characteristics of the problem. So both tools may be used in combination or used separately if the approach is more suitable. The development iteration will last for a period of two to three weeks where the general development plan from the planning session will act as a scrum backlog. Each problem to be solved will be on the backlog and part of the design for the solution will be pre-determined from the iteration planning session. The developed solutions will be deployed to a test environment where the relevant Atlassian products are set up as if it was the workstation belonging to one of the Atlassian administrators.

4.5 Evaluation and reflection

After each iteration there will be a evaluation and reflection session.

The evaluation session will be used for testing the newly developed solutions on the Atlassian administrators. These tests will not only serve as an acceptance test of the solution
4.5. Evaluation and reflection

features, but also as a benchmark of the amount of necessary steps. The evaluation method is to measure the amount of manual steps necessary to perform the benchmark test task, it can be usage of keyboard keys and mouse clicks. If the new benchmark values are improved, compared to the first original benchmark values, it means that the new solution was a step in the right direction. There will hopefully be a significant improvement of benchmark values after the final iteration is completed. The solutions that lead to this improvement will be part of the answer to the research question in the introduction chapter.

Before moving on to the next iteration there will be a reflection session of the current iteration, to learn from previous mistakes and improve the next iteration.
This chapter will present the results that was produced by following the method.

5.1 Pre-study

The pre-study of the Atlassian plugin SDK and REST API revealed the following facts:

**Atlassian plugin SDK**

1. Can be used to create a so called add-on for Atlassian Jira, Confluence, or Bitbucket
2. The add-on will add a new layer of functionality on top of the existing product functionality
3. The plugins can be added to each of the Atlassian products through a plugin manager by users with administrator permissions
4. Can make calls to the REST API in the plugin source code
5. Appears to be quite complex to use and requires training time

**Atlassian REST API**

1. Can gather or update the information in the fields in the elements in each Atlassian product
2. Good for scripting purposes as it relies a lot on JSON formatted messages
3. Can not modify the appearance or functionality of the Atlassian products like the plugin SDK
5.1. Pre-study

The pre-study shows that while the Atlassian REST API is very flexible in terms of usage, it is also restricted to manipulation of information. Depending on the issue at hand it may be more suitable to develop a solution using the plugin SDK, which has more solution opportunities.

The initial interview revealed the common obstacles that the Atlassian administrators usually bump into when they perform their daily work tasks. As expected it was mainly repetitive tasks that involve a lot of mouse clicks and navigation back and forth between different GUI pages. Frequent audits of access permissions are performed in all three Atlassian products to cope with the requirement of high IT security at Saab Group AB. The execution time and the frequency of these permission audits are heavily affected by the issue described above. The following issues were revealed about each of the Atlassian products during the initial interview:

**Jira**

All the Jira projects must go through a reoccurring permission audit where the Jira administrator will check if the correct user or user-group has permissions in the project. This is a tiring process since there may be hundreds of projects to verify where the permission details are on a separate GUI page for each project. This means that repetitive page navigations will be performed where the Jira administrator may overlook important permission details of a project.

Sometimes the Jira administrator needs to dive deeper into the audit of a project to investigate suspicious groups or users connected to the project. This requires quite a few extra clicks at the moment and is more time-consuming than it has to be.

Sometimes it is easier to clear out confusions during the permission audit process simply by checking the change-logs of the project permissions. This is however not that simple since the logs contain the changes made to every existing project, and any kind of change made to a project is logged. Navigating to the log page requires quite a few clicks at the moment and finding the correct buttons to click may be difficult.

**Confluence**

All the Confluence spaces has to go through the same reoccurring permission audit just like in the Jira case. The same problem exists here, where the Atlassian administrator has to navigate to a specific GUI page, in each space, in order to see the access permissions for the specific space. This can be done in a more efficient way.

Sometimes there may be individual users who gain special access permissions to a space without being in one of the allowed user-groups. These users should not be able to see the space at all and causes a security problem.

**Bitbucket**

The reoccurring permission audit process are performed on the Bitbucket projects as well and suffer from the same repetition problem as Jira and Confluence. The Bitbucket administrator has to navigate through several GUI pages in each project to be able to see the access
permissions.

One of the common tasks in Bitbucket is creating repositories in a project, sometimes several repositories with very similar settings. This is of course very time-consuming and it is very easy to overlook a step when configuring the settings in each repository. The settings are quite complicated to set and several add-on settings need to be configured as well.

Sometimes the Jira administrator may need to reconfigure the same setting on several repositories, this is not possible at the moment and the process is time-consuming.

5.2 Benchmark test

The initial interview clearly showed that the most time-consuming tasks are the audit process of permissions, something that occur in all three of the products. There may be sometimes hundreds of projects to audit which would take a long time to finish with the current solution and several errors may be committed by the auditor. Something that would make the audit process more efficient is checking the logs and search for project permission changes. This is however also time-consuming since the log files will log any change made to any project in all three of the Atlassian products. The benchmark test will examine how many steps/clicks that would be necessary to audit a certain amount projects in each of the Atlassian products. It will also test the amount of necessary steps required for checking the logs.

Audit process:

1. **Jira**: 6\*N clicks, where N is the amount of projects
2. **BitBucket**: N\*6\*(1+M(N)), where N is the total amount of projects and M(N) is the average amount of repositories in all the N projects
3. **Confluence**: N\*5 clicks, where N is the amount of projects

Checking logs:

Currently undefined since the mix of log updates will make the task very demanding when it comes to clicks. For now any change to the process of checking the logs will be considered as an improvement.

5.3 Iteration 1

Plan

The first issues that will be taken care of is the issues connected to the audit process in each of the Atlassian products. The Atlassian plugin SDK will not be used in any of the iterations due to its complexity and the learning curve for learning how to use it properly. The solution will instead be developed by using python scripts in combination with the Atlassian REST API which is easier to understand and make use of.
Problem analysis

Jira and BitBucket audit issue

This problem analysis covers the audit process in Atlassian Jira and BitBucket.

1. It is possible to solve the problem by using either the rest API (python script) or the plugin SDK
2. There is no need to divide the problem into further smaller problems
3. This problem is worth fixing and the required time is within the iteration time window
4. The REST API is best suited for solving this problem
5. The idea is to use HTTP calls to the REST API in order to gather the project information of all the projects and present it as a summary in a local file
6. The solution shall present a summarized view of the project permissions from all the projects in a readable format

Confluence audit issue

This problem analysis covers the audit process in Atlassian Confluence.

1. It is possible to solve the problem by using the Atlassian plugin SDK
2. There is no need to divide the problem into further smaller problems
3. The solution is not worth implementing because of the training time required for learning to understand the workings of the Atlassian plugin SDK
4. The REST API is best suited for solving the problem due to the limited time frame
5. The python script will use the Confluence REST API to collect the permissions information of all the projects and print it to a local file
6. The solution shall present a summarized view of all the permission information from all the Confluence projects in a readable format

Development work

Jira

The issue was solved by developing a python script that makes use of the REST resource "/rest/api/latest/project", which is the go-to resource for project related information in Jira. The "latest" section in the resource will point to the most recent version of the REST API and can be replaced with i.e. "/2/" for version 2 of the REST API. There is an example of the JSON output in figure 5.1. The first part in the url, "(https://localhost:8080/)" is the base-url where Atlassian Jira is running and the REST resource above is appended after the base-url. Parts of the result is hidden to lower the overhead and limit the amount of information presented in the result. Adding i.e. "?expand=description" as a parameter after the REST resource will add the project descriptions to the result. There are several sub-resources under the "/rest/api/latest/project" resource that can be used for more specific project information such as project permissions and project roles. Figure 5.2 displays an
example output of the more specific permission information for each project role, the sub-resource "/{projectID}/role/{roleID}" was used for this. The main Python library that is used is the "http.client" library, which allows users to make HTTP method calls (get, post, put and delete) to a REST API. The main reason for using this library is that it is part of the standard python package, which means that no external library files is necessary. The python script uses dictionaries created from the JSON messages to iterate over the collected information.

Confluence

It would be possible to implement a solution to the Confluence issue if the Atlassian plugin SDK is selected as approach, due to the lack of functionality in the Atlassian REST API. This activity would however consume too much time since most of the time would be spent on learning how to use the plugin SDK. One different approach would be to use the deprecated Atlassian SOAP API, but this is not a long term solution and it requires external python libraries which Saab won’t allow in the systems. This issue will be left unresolved until the Atlassian REST API is upgraded to a version that can offer the necessary functionality.

BitBucket

The problem was solved by using the same kind of solution as in the Jira case, the REST resource that was used in this case was "/rest/api/latest/projects" and the base url is "(https://)localhost:7990/". This REST resource will return a JSON message containing the project keys of all the existing BitBucket projects, example in figure 5.3. The REST sub-resources "/{projectkey}/permissions/groups" and "/{projectkey}/permissions/users" was used to gather more specific permissions information for groups and users, example in figure 5.4. The main difference is that every Atlassian BitBucket project contains one or more repositories with permissions settings that is of interest to the BitBucket administrator. The REST sub-resource "/{projectkey}/repos/{repositorykey}/permissions/group" and "/{projectkey}/repos/{repositorykey}/permissions/users" was used to collect this information, example in figure 5.5.

Evaluation

Jira

The new solution will gather all the necessary project permissions information by using the Jira REST API and present a JSON summary of the access permission information from every project in a separate text file, see figure 5.6 for a visual example. The Jira administrator can launch the python script in a command prompt which will ask for the Jira login credentials and then create the text file containing the information summary.

This approach basically presents the same information as the usual Jira permissions GUI page but requires far less steps than before. In the old solution, the Jira administrator would have a GUI page listing all the existing projects. To actually reach the permissions information in each of these projects requires a total of 3 clicks, going back to the list of projects requires another 3 clicks. So a list of N projects would require a total of $6N$ clicks as opposed to just the few clicks in the new solution that was necessary in order to start the script and open the generated text file.
BitBucket

The new solution is very similar to the Jira solution above with the exception of the repository category in every project, otherwise the same approach has been used, see figure 5.9 for visual example. The BitBucket administrator can launch the script in a command prompt which will ask for BitBucket login credentials and create a separate text file with the project permissions summary.

The BitBucket GUI pages has the same kind of list as in the Jira case, containing every existing BitBucket project. Reaching the actual GUI page containing the project permissions info requires 3 clicks. Going from this page to the GUI page containing the list of repositories requires 2 clicks and going from this page to every repository permissions settings page requires 3 clicks. The total amount of clicks required for auditing all the projects would in the end be \( N \times 6 \times (1 + M(N)) \), where \( N \) is the total amount of projects and \( M(N) \) is the average amount of repositories in all the \( N \) projects. This new solution only requires a few steps (as in the Jira case) while the existing solution requires a lot of steps even when the amount of projects \( N \) is small.

Reflection

Many of the problems during the iteration was lack of compatibility and functionality. The python script can’t rely on external libraries which sometimes was a requirement for solving the issues at hand. Other times it was the lack of functionality in the REST API, which meant that the only feasible solution was to use the Atlassian plugin SDK, which is not an option due to the limited time frame.

5.4 Iteration 2

Plan

The next issues that will be dealt with is checking changes in the logs more efficiently, both in Atlassian Jira and BitBucket. The possibility of performing in depth audits of users will be examined as well.

Problem analysis

This problem analysis covers the issue with checking changes in the Jira and BitBucket logs.

1. It is possible to solve the problem by using either the rest API (python script) or the plugin SDK
2. There is no need to divide the problem into further smaller problems
3. This problem is worth fixing and the required time is within the iteration time window
4. The REST API is best suited for solving this problem
5. The idea is to use a python script that compares the current project permissions with the project permissions that has been preserved in a local log file.
6. The solution shall present a summarized view of the changes that has been made to the project permissions in every project in a readable format

This problem analysis covers the issue with in depth investigations of specific users.
1. It may be possible to solve the problem by using the Atlassian plugin SDK
2. There is no need to divide the problem into further smaller problems
3. The solution is not worth implementing because of the training time required for learning to understand the workings of the Atlassian plugin SDK
4. The best solution would be to use the REST API due to the time limitation
5. The solution would use Python scripts to list all the users in each project based on their internal or external status
6. The solution shall present all the internal and external users in a summarized readable view

Development work

Jira

The new solution to the Jira audit process from iteration 1 will be merged into the new solution to the issue regarding changes in Jira permissions. Both of these solutions will be put into the same tool where the user can select an option to execute, figure 5.7 shows an image of this menu. The solution to the issue in this iteration will use a log file to compare with the current project permissions. The user can input a specific log date to compare with, so it is possible to go back several weeks if necessary. This comparison will generate a local diff file that will point out the changes that has been made to the permissions in every project, figure 5.8 shows an example of this.

The possibility to perform in depth investigations of specific users will not be implemented due to the lack of functionality in the REST API and the requirements of the functionality. The Jira administrators sometimes wants to see if a user is a so called external or internal user, something that currently is not supported in the Jira REST API.

BitBucket

This solution is more or less identical to the one in the Jira case, there is a command prompt menu as in figure 5.7 and a comparison file with logged entries in it, see figure 5.9 for visual example. The local diff file is generated the same way as well and will point out all the fields that has been modified in each project, see figure 5.8 and 5.10 for visual example. The BitBucket REST API lacks the functionality to detect if a user is external or internal so this issue will be overlooked for now.

Evaluation

Jira and BitBucket

The new solution will improve the process of auditing Jira or BitBucket project permissions even further. Now the administrator can compare previous project permission settings dating back months or even years if necessary. All of the new solutions are gathered under the same menu (different sub-menu for Jira and BitBucket though) and the user can choose what to do simply by entering a number. Checking the logs in the existing solution
would require several steps and finding the correct log entry that contains the changes made to the specific project would require further steps. The decrease in required amount of steps is significant in both the Jira and BitBucket case as compared to the original solution.

Reflection

This iteration was more fluent since more thought was put into compatibility and functionality of the Jira REST API. Problems that required external Python libraries or non-existing REST API functionality was overlooked early on.
5.4. Iteration 2

Figure 5.1: An example of the JSON output from the Jira REST API.

Figure 5.2: An example of the role specific JSON output from the Jira REST API.
Figure 5.3: An example of the JSON output from the BitBucket REST API.

Figure 5.4: An example of the JSON output from the BitBucket REST sub-resources "/{projectkey}/permissions/groups" and "/{projectkey}/permissions/users".
Figure 5.5: An example of the JSON output from the BitBucket REST sub-resources 
"/{projectkey}/repos/{repositorykey}/permissions/group" and
"/{projectkey}/repos/{repositorykey}/permissions/users".
5.4. Iteration 2

Figure 5.6: An example of the text file containing the summary of Jira project permissions. "TEST" & "TEST2" is the projects and "group" & "user" is the user groups or users that has been added to each role in each project. Each role has different permission settings.

```json
12/05/2017
{
    "TEST": {
        "roles": {
            "Administrators": {
                "group": [
                    "jira-administrators(jira-administrators)"
                ],
            },
            "Developer": {
                "group": [
                    "jira-administrators(jira-administrators)"
                ],
                "user": [
                    "user1(user1@user.com)"
                ],
            },
            "Normal user": {
                "group": [
                    "jira-administrators(jira-administrators)"
                ]
            }
        }
    },
    "TEST2": {
        "roles": {
            "Administrators": {
                "group": [
                    "jira-administrators(jira-administrators)"
                ]
            },
            "Developer": {},
            "Normal user": {}
        }
    }
}
STOP
```

Figure 5.7: The tool menu.
Figure 5.8: An example of how the Jira diff file works, the info marked green has been added and the info marked red has been removed from the latest permission settings as compared to the chosen log entry 22/05/2017.
Figure 5.9: An example of how the BitBucket comparison file could look like. "TEST2" is the project key and "group" & "user" contains all the user groups or users that has been added to a specific permission. Other details such as if the project/repository is public is visible too.
Figure 5.10: An example of how the generated BitBucket diff file could look like. Figure 5.8 gives a detailed explanation of how it works in the Jira case.
6 Discussion

The results and method of the thesis will be discussed in this chapter.

6.1 Results

The results of the work are based entirely on python scripts, REST calls, and JSON formatted text files (requested by Saab), yet the performance/efficiency are increased. Doing all of these permission audits would normally be considered annoying, even when the info is described in a colorful GUI page, simply because of all the clicks and page navigations. The theory that automating simple repetitive tasks could save so much effort for the operator was confirmed, even though the visualization of the results are in JSON text format. One would expect that colorful GUIs would be required in order to increase the efficiency of something like this but in this case it was not a requirement.

6.2 Method

The method that was followed was the usual iteration based development life cycle with key activities such as planning, development and evaluation of the implemented code. This approach was good since every iteration was at most two weeks long and changes in the process could be made in between iterations after a session of reflection. What was less good was the evaluation activities since they were based mostly on the amount of required steps/mouse clicks. This is very easy to measure and is very reliable when comparing different solutions of a problem, but it is not a good measure of usability or efficiency. One can often assume that less steps from the operator automatically means more efficiency but it was difficult actually measure this. But since the same information is presented in the new summary-based solution as in the old GUI solution, it is safe to assume that the new solution delivers all of the information faster and more summarized than before. So in this case the step-based evaluation was a good choice for measuring potential improvements in efficiency. It would had been interesting to see if the combination of the REST API and the plugin SDK could had delivered a better solution. But the learning curve of the plugin SDK was a big (unexpected) limitation, since the thesis project had a limited time frame.
6.3 The work in a wider context

The thesis work has presented a new way to perform a otherwise time-consuming task in a more efficient and less demanding way. Tasks such as this project-access permission audit is very demanding and should always be performed in the least operator demanding way possible. The current state of the solution only uses rather simple text files to present information but can easily be improved in the future.
Conclusion

The whole purpose of the thesis work was to investigate how customized solutions could be applied to the Atlassian products, to increase the efficiency, by decreasing time usage of common tasks. The tasks that was demanding in this case was in fact only demanding because of the repetitive actions and GUI page navigations, something that easily can be solved through automation. Even though the solution isn’t as fancy looking as the existing GUI solution, it still saves so much effort compared to before.

So as a conclusion, the answer to the research question:

**How should the customization to the Atlassian products be designed to achieve minimal time usage for common tasks?**

is python scripts combined with the Atlassian REST API. The result of the thesis work proves that simple automated scripts can achieve a lot compared to a detailed GUI page. The same result can of course be achieved through an add-on in the GUI pages, but require a lot more development efforts compared to scripting.

As a future improvement, the resulting JSON text files can be displayed in a more readable, perhaps more colorful, way in order to improve the usability of the new solution. There are a lot of JavaScript frameworks available online that can use JSON data and display it in various ways depending on the requirements of the user. It may also be possible to use the script solution in an add-on, developed through the Atlassian plugin SDK.
Bibliography


[9] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. “Semantic Web Services”. In: IEEE Intelligent Systems 16.2 (Mar. 2001), pp. 46–53. ISSN: 1541-1672. DOI: 10.1109/5254.920599 URL: http://dx.doi.org/10.1109/5254.920599.


