Bachelor's Thesis

Evaluation of Different Project Management Approaches in Software Projects in a University Context

Evaluierung von verschiedenen Projektmanagementansätzen in studentischen Softwareprojekten

Max Plaga

Hasso Plattner Institute at University of Potsdam

July 31, 2019

Bachelor's Thesis

Evaluation of Different Project Management Approaches in Software Projects in a University Context

Evaluierung von verschiedenen Projektmanagementansätzen in studentischen Softwareprojekten

by

Max Plaga

Supervisors

Prof. Dr. Christoph Meinel, Christian Tietz, Eric Klieme

Internet-Technologies and Systems Group

Philipp Berger, Stephan Schultz, Uwe Leppler

neXenio

Hasso Plattner Institute at University of Potsdam

July 31, 2019

Abstract

This thesis takes a look at prominent project management approaches for software projects and evaluates their usefulness in the special context of student projects. To do so it evaluates each approach with a top-down approach using known project management success criteria. These are stakeholder satisfaction and efficiency of the development cycle. This top-down angle is afterward complemented with a bottom-up analysis using interviews of different bachelor's project teams and one industrial project for comparison. After evaluating different approaches a comparison is drawn that is aimed at helping future university projects to choose the right project management framework and avoid common mistakes.

Zusammenfassung

Diese Arbeit betrachtet häufig benutzte Projektmanagementansätze für Softwareprojekte und evaluiert ihre Nützlichkeit in dem besonderen Kontext von studentischen Projekten. Um dies zu tun wird jeder Ansatz mit einer Top-Down und einer Bottom-Up Evaluierung untersucht. In dem Top-Down Ansatz werden bekannte Erfolgsfaktoren von Projektmanagement einbezogen. Für den Bottom-Up Ansatz hingegen wurden mehrere Teams von Bachelorprojekten und ein industrielles Projekt befragt. Nachdem die verschiedenen Ansätze evaluiert werden, wird eine Abwägung angefertigt, die zukünftige studentische Softwareprojekte bei der Wahl des richtigen Projektmanagementansatzes untertstützen soll.

Contents

1. Introduction **1**
1.1. Motivation 1
1.2. Project Scope 1

2. Background **5**
2.1. Terminology 5
2.2. Tools 6
2.3. Interviewed projects 6

3. Related Work **9**
3.1. Summaries 9
3.2. Conclusion of related work 10

4. Evaluation Framework **11**
4.1. Differentiating student projects from industrial projects 11
4.2. Setting the evaluation framework 13

5. Evaluation **17**
5.1. Laissez-faire 17
5.1.1. Top-down evaluation 17
5.1.2. Bottom-up evaluation 19
5.2. Lean Project Management with Kanban 20
5.2.1. Top-down evaluation 21
5.2.2. Bottom-up evaluation 23
5.3. Iterative development with Scrum 24
5.3.1. Top-down evaluation 25
5.3.2. Bottom-up evaluation 27
5.3.3. Further insights from project "BAAM" 28
5.4. Comparison 31
5.5. Limitations 33

6. Conclusion and future work **35**

A. Appendix **37**

References **41**

1. Introduction

Benjamin Bloom presented six levels of progressing complexity of reasoning skills needed in classrooms in ″Bloom′s Taxonomy″ [1]. The higher the level the students are able to apply the deeper their understanding of a topic he concludes. Bloom lists ″Knowledge″ as the first level while ″Application″ is already the third level. That is why most university programs include practical projects in which students can apply and consolidate their theoretical knowledge.
This chapter will justify the usefulness of this thesis and explain the scope and objective of the project this paper resulted from.

1.1. Motivation

A project like a bachelor′s project is often the first time students are confronted with a project of this magnitude. In smaller projects, students can get by without proper project management. But in a project with such a scope students need to plan ahead carefully and need to represent their project to external stakeholders. For this to work they need to start thinking about how to manage their project proactively. There are multiple different methods and frameworks that aim to help the project management of software projects. A student team now faces the complicated task to choose the one that is the most appropriate for the undertaking. When looking for guides on which framework to choose they often find themselves reading about project management in industrial projects and not something specific for their needs in a university setting.

This thesis will take a look at different methods and frameworks that are commonly used in project management in software projects and evaluate them based on their appropriateness for student projects. It will also try to prevent future student projects from making the same mistakes as past projects by evaluating already existing projects.

1.2. Project Scope

In our project ″Behavioral Authentication Access Management″ the goal was to make physical access control more secure while sparing hardware costs. Physical Access Management is a field where security is crucial. Company premises need to be reliably protected to keep intruders from gathering valuable company data or harming them in any other way. As of now facilities can use a variety of different methods to try to prevent break-ins. Entry gates can be equipped with diverse identification hardware.

Each of them has their own drawbacks; lacking security or high costs being the most prominent ones. Iris scanners count to the most secure options but are expensive. Access Badges, on the other hand, do not cost as much money as iris scanners but are a risk factor in case of loss.
A new way of authenticating someone is by their behavior. Our project partner neXenio implemented a machine learning algorithm that identifies a user by special traits like their gait or the way they pull out their phone. For this new technology to be used in a practical setting we built a platform on which an employer/user can register and afterward link one or multiple smartphones. The user's smartphone will be used as a security token by the gates to verify their identity. Afterward, employers can use our platform to grant access rights to their employees. This can be done by granting them access to certain gates directly or indirectly by adding them to customizable groups like "interns" or "management". Behavioral Authentication in combination with the user's smartphone allows facility managers to save money on securing entry gates and increases the level of security at the same time by adding another factor for authentication. It will also enable employees to seamlessly enter restricted areas. This means that they will be able to simply walk towards a gate and it will open when they are close if they are authorized to pass it. And all of this without presenting any additional form of ID.
The built system in this project also has a feature to control a gate's needed *trust level* for passing. The *trust level* is what is put out by our project partner's "Behavioral Authentication" application. Matching behavior is not a *true* or *false* question like a lock's security pins or a password. Instead, a more nuanced result is given. The *trust level* says on a range from 0-100 percent how certain the smartphone application is that the person in possession of the device is the rightful owner. The feature to control the *trust level* needed to pass a gate was introduced due to the fact that facility managers may not require as much security for some areas like a cleaning supply room but want to be absolutely sure only authorized personnel enters other areas (e.g. only server admins enter the server room). The *trust level* for the server room could be set higher than that of the cleaning supply room through our system. As of now different areas in a facility usually have different security mechanisms with varying levels of security too. With the "Behavioral Authentication Access Management" system these varying levels of security can still be achieved while unifying the hardware needed to do so.

To reach our goal to build this system as efficiently as possible we decided in favor of some project management approaches while refraining from using others. This thesis will take a look at the different approaches. The evaluation of those will show why some of them work better than others in the context of student projects.

Structure

This thesis will be structured in the following manner: first it will introduce the terms needed to follow along and the tools this thesis will use to analyze different approaches in **Background** (chapter 2). Afterward it will take a look at other studies in **Related Work** (chapter 3) to see how much of the already available information can be applied to this context. In **Evaluation Framework** (chapter 4) it will explain how the different methodologies can be distinguished in a quantifiable way. After setting the evaluation parameters this thesis will pick multiple approaches and evaluate them in the context of software projects by students in **Evaluation** (chapter 5) before drawing a **Conclusion** (chapter 6).

2. Background

This chapter will lay the background for the content to come. At first it will introduce some terms and define them in the context of this thesis. Afterwards it will take a look at the tools used for chapter 5: Evaluation. In section 2.3 it will show the main objective of selected software development projects. These will later be used to aid the evaluation in chapter 5.

2.1. Terminology

BAAM In this thesis "BAAM" will be used as an abbreviation for "Behavioral Authentication Access Management" , a new way of managing physical access, but also for the name of our bachelor's project. Our project will be used for real-world analogies and findings.

Development Cycle The development cycle is the whole workflow of getting one feature from being requested to deploying it in production. The time this takes is often called *cycle time*. In Kanban *cycle time* is a good metric to see how fast development is progressing.

Agile Methods "Agile is the ability to create and respond to change" [2]. Agile software development uses methodologies and frameworks like scrum, XP or Feature-Driven Development. Collaboration and the self-organization of teams play a big role in the agile software development community. Being agile is about more than just applying frameworks, it's about the core values and principles behind them.

User Story A feature request that is written from the perspective of a user. As they are written from user perspective they are not tied to the implementation and describe a requirement in natural language. A common pattern for a user story is *As a [person] I want to [...] because [reason].*
Two of the user stories from our project "Behavioral Authentication Access Management" were "As a facility manager I want to be able to allow certain users or groups to pass a specific gate." and "As a company admin I want to be able to import user data from a CSV[1] file because my company does not have an LDAP[2]".

Story Points A value that is given to *User Stories* to make them comparable in terms of effort needed to fulfill them. One thing to keep in mind is that story points do not directly reflect units of time but can have arbitrary values like "*3 pieces of*

[1]Comma-separated values
[2]Lightweight Directory Access Protocol

cake" or "*8 monsters*". The development team agrees on a sequence beforehand. Our team worked with values on the *Fibonacci sequence* ([1, 2, 3, 5, 8, 13, 21, . . .]) as it was easier to differentiate between a "5" and an "8' than between a "5" and a "6". Also the Fibonacci Sequence accommodates the fact that bigger stories are usually harder to estimate than smaller ones. This thesis will use "SP" as an abbreviation for *story points*.

Student project A project undertaken by students. For this thesis we will view a *student project* as a project for which students receive a grade as part of their studies. In our context a *student project* is usually accompanied by supervisors from the university. The requirements and challenges are set by an external project partner. In some cases this project partner is more of a customer in other cases they work together as one team to build something that is going to serve someone else.

2.2. Tools

JIRA An issue tracking product developed by Atlassian[3] that allows bug tracking and agile project management. JIRA's charts plugin can be used to visualize sprint progress to make sure the development team is staying on track. Those charts will be used in this thesis to analyze different development approaches.

2.3. Interviewed projects

For this thesis several software projects were interviewed to gain insights into what project management approaches they follow. The results of the interviews can be used to find common pains and see if choosing a different approach would have helped prevent some of them. This section will briefly introduce the different projects and explain their challenge.

BAAM (see section 1.2)

HPI School Cloud This bachelor's project's goal was to make commercially licensed study material available in the online learning store of the HPI school cloud.

EPIC The aim of this bachelor's project was to help case workers in companies by finding easy and often repeated tasks in an ERP[4] system so that they can later be automatized.

Fighting Outbreaks The bachelor's project team of "Fighting Outbreaks" built a system that aims at increasing the efficiency of DNA analysis to fight outbreaks in real time.

[3]`https://www.atlassian.com/software/jira`
[4]Enterprise-Resource-Planning

Kyub Kyub is a system built by one bachelor's project that shortens the time until a physical prototype is ready by transforming three-dimensional models to laser cuttable planes.

bdrive This industrial project was used as a reference. Their goal is to build a secure cloud storage "Made in Germany" for highly critical data.

3. Related Work

If I have seen further than others, it is by standing upon the shoulders of giants.

Sir Isaac Newton

Before starting an own evaluation this thesis will study related work and see to which degree their findings can be used for this paper.

3.1. Summaries

Comparing Alternative Software Development Life Cycle Models

In their paper ″A strategy for comparing alternative software development life cycle models″[3] Davis, Bersoff and Comer introduce a framework to compare alternative life cycle methods for software development. Their understanding of ″alternative″ includes most methods that are not the *waterfall model*. So it takes a look at *rapid throwaway prototyping*, *incremental development*, *evolutionary prototyping*, *reusable software* and *automated software synthesis*. For each method it analyzes their Shortfall, Lateness, Adaptability, Longevity and Inappropriateness. It then compares these criteria in graphs that visualize the development life cycles for different approaches.

Success dimensions of international development projects

Diallo and Thuillier were one of the first to do empirical research on the specific success criteria and factors of international development projects. They published their findings in their paper ″The success dimensions of international development projects: the perceptions of African project coordinators″[4].
As software engineering is probably the branch of engineering that is most easily outsourcable and combinable with remote work software projects inevitably reach international dimensions.

Project Feasibility: Points of Vulnerability

The book ″Project Feasibility: Tools for Uncovering Points of Vulnerability″ [5] by Mesly offers tools that will help decide if a project should be continued, altered or canceled. It does that by analyzing the points of vulnerability of a project. Mesly sets a new focus on ″identify[ing] ′hidden truths′ and eliminat[ing] those gray areas that jeopardize the success of a given project ” [5]. The book ”Project Feasibility” suggests inspecting a project for vulnerabilities in the four aspects: People, Power, Processes and Plan. According to the book these points come together and can each bring positive or negative forces. Only when the positive outweigh the negative forces can a project succeed. To support the argumentation on what Mesly calls ”the four Ps” he takes three examples: the construction of the Montréal Olympic Stadium, Québec Multifunctional Amphitheatre and Mervel Farm project. The three projects with a big range of budget and size are then evaluated with the four Ps in mind.

Case Study: Student collaboration in software development

In their work ”Student Collaboration across Universities: A Case Study in Software Engineering” Brereton et al. [6] report on work undertaken by students in their final (third) year project. They argue that students are usually only part of projects that consist of peers from the same university with the same background. In contrast they focus on projects that require a collaboration of students of different universities. This work then focuses on low-cost tools that can be used for distributed collaborations. In their evaluation Brereton et al. evaluate the administration, software engineering issues and technologies of the student projects.

3.2. Conclusion of related work

A lot of research has been published that investigates past projects and their project management approaches [4, 5]. Equally as much research exists for project management in software development [3]. Other researchers set a focus on student projects like Brereton et al. [6]. They even look at student projects that develop software. But the focus is not on their project management. As of the writing of this thesis not much evaluation effort has gone into the intersection of all these aspects. Software projects by students are where those three aspects overlap. Since those projects have individual challenges further research should be conducted on it.
To some extend this thesis will still be able to benefit from past research. To be specific the related work mostly influenced the decisions on how different managment approaches will be made comparable. Davis et al. [3] and Mesly’s [5] most valuable work for this paper will be their evaluation framework itself as those points will be taken into consideration in chapter 5 when taking a look at different methods and frameworks with the focus of this thesis in mind.

4. Evaluation Framework

To make this thesis's evaluation as quantifiable as possible it needs to set an evaluation framework. This chapter will break down what factors go into an argument for or against an approach in the special context of student projects. To be able to rank the different points to some extend based on their importance for student projects this thesis will first differentiate student led projects from industrial projects. That will help in deciding if some factors play a more significant role for student projects than others.

4.1. Differentiating student projects from industrial projects

When trying to differentiate between software projects by students and those in an industrial context several key differences come to mind. One of the major ones is that industrial projects have money on the line. Project managers need to act accordingly to stay on schedule and budget. The PMBOK lists managing "competing demands for: scope, time, cost, risk, and quality" [7, p.6] as one of the main tasks of project management. When taking a look at the five brought up factors in reality *cost* and *risk* are usually not the center of attention in a university project. In industrial projects, on the other hand, they play a vital role. Going over budget is a common problem in IT projects as seen in Figure 4.1. This oxford study of 5,400 large scale IT projects (defined as those with initial price tags exceeding $15 million) found that on average, large software projects run 66 percent over budget and 33 percent over time, while delivering 17 percent less value than predicted [8].
Another big factor is that the students working on university projects have different

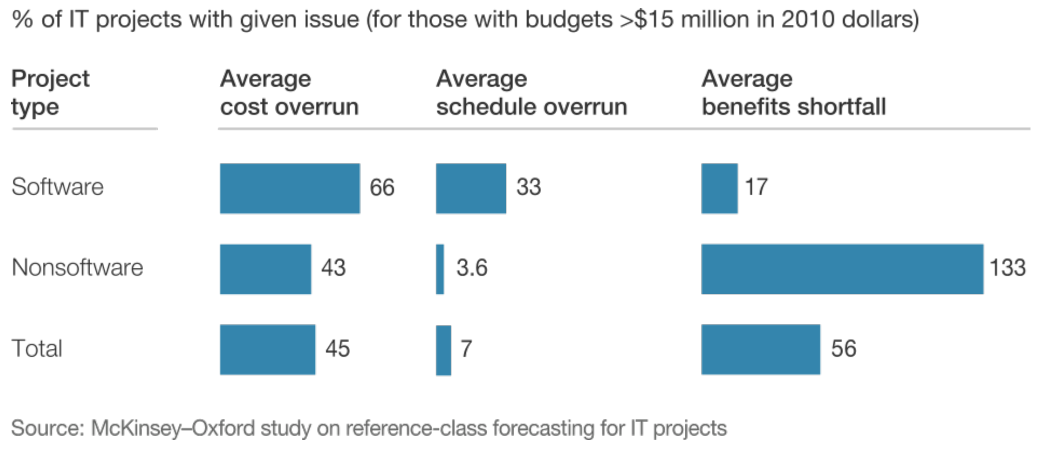


Figure 4.1.: Oxford study on reference-class forecasting for IT projects

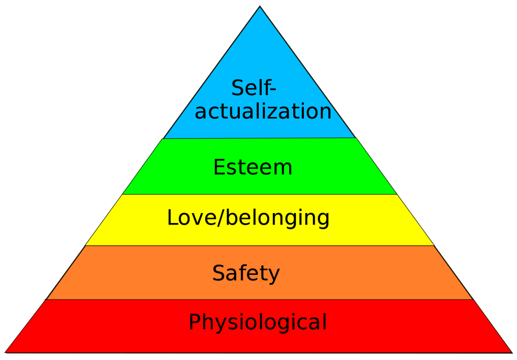


Figure 4.2.: Maslow's hierarchy of needs[5]

motivational factors than regular employees. For one they do not get compensated monetarily. So their motivation needs to come from other factors. On the other hand students often receive a grade for their efforts. They also have the opportunity to gain valuable knowledge for their future endeavors. When the project partner does not set the parameters for students to learn they risk the project outcome by lowering the team member's motivation significantly. One student commented that "good requirements management causes good motivation in the team". As a bonus in a setting of a student project students can start building lasting relationships with potential employers or mentors. This can work as a motivation for students to outperform their peers and excel in front of their mentors. In Maslow's hierarchy of needs the need to be respected by others falls in the level of "Esteem" - the second highest level (Figure 4.2). But as mentioned this could also lead to a position in the company later which could act as additional motivation.

Another difference is that the supervision of a student team is usually not as extensive as that of a regular team no matter what project management approach is being followed. This could be since no money is on the line.
Not only the supervision while working on the project but also the onboarding process in the beginning could suffer from companies not having to loose as much with student projects. With regular employees companies want to make sure their workers are ready to provide value as soon as possible. Hence a lot of companies have a thorough onboarding process that introduces new employees to the tools and practices used. In the interviewed bachelor's projects that onboarding process was not really extensive and some students felt that they would have been able to provide valuable work sooner if they had received more guidance in the beginning. The projects that started from zero instead of expanding an existing system also supposed that they would have profited if the project partner had set up some of the infrastructure in advance. One suggestion was to let experts of the partner set up a testing environment and integrate it into the

[5]https://en.wikipedia.org/wiki/Maslow%27s_hierarchy_of_needs

delivery pipeline before starting the project. Those experts should be able to complete the process in a shorter period of time as they must have already done it for other projects. With the pipeline already set up the students can dive right into the project and focus on providing value.
One reasons for not setting up a working environment in advance could be to grant the team freedom and let them choose their own components. Another could simply be that the project partner does not want to use paid resources to set up an environment for a project that does not generate revenue.

Differences in the level of expertise of the teams are also prominent. Team members of student projects might work in a practical project of this magnitude and duration for the first time. Not every curriculum includes several software projects which each span a whole semester or multiple months. It's no doubt that industrial teams bring more senior experience into a project. Data provided by The United States Census Bureau[6] can be aggregated to find that the average age of software developers in the USA was 39.8 years in 2017 [9] which statistically means that they each had at least 10-20 years experience after university. With the difference in expertise levels choosing a project management approach should probably be under different angles as well (see chapter 5).

According to the "on-site customer" principle in XP someone who really uses the system should be on-site and a part of development at all times. This is done to shorten the development cycle by providing the team with someone to answer questions as soon as possible without having unnecessary waiting times. In a student project that is a joint operation between the university and an industrial partner an on-site customer may not always be available. This could increase the time until important questions are answered and bring the cost of context switches.

4.2. Setting the evaluation framework

Top-down evaluation using success criteria

First each approach will be looked at under a top-down approach. For this known project management success criteria will be taken into consideration. These are:

Stakeholder Satisfaction

According to the PMBOK Guide [7] project stakeholders are

> *Individuals and organizations that are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or project completion; they may also exert influence over the project and its results.*

[6] `https://www.census.gov/`

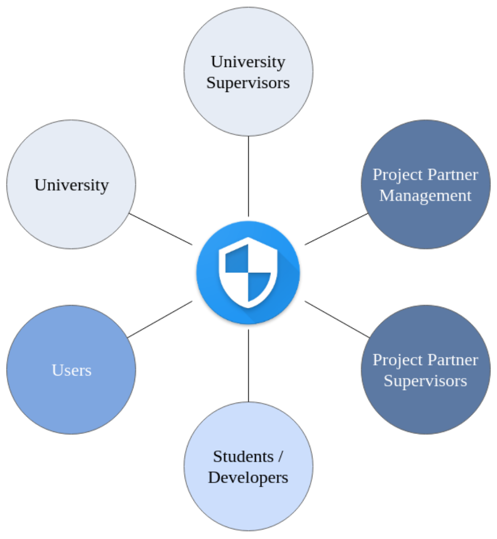


Figure 4.3.: Stakeholders of our project "Behavioral Authentication Access Management"

As the PMBOK shows the project's stakeholders have an interest in the outcome of the project. Therefor it is always a good idea for a project team to keep in mind who their stakeholders are as those will be the people that need to profit from the project. This evaluation will take into account how likely it is that the stakeholders will be satisfied with the project outcome. The stakeholders of our project "Behavioral Authentication Access Management" can be seen in Figure 4.3 for reference. As exemplified in this project the stakeholders can come from various backgrounds. Especially their technical knowledge can vary drastically so that needs to be considered as well. Users do not care about what is powering an application in the background or its performance as long as it does not affect them negatively. Goodhue concluded in [10, p.1840] *"[...] the value of a technology does appear to depend upon the task of the user. Users seem to view their system as tools which assist or hinder them in the performance of their tasks."*.

The interviewed projects (see section 2.3) all had at least the following list of stakeholders:

1. Project partner (management and supervisors combined)
2. University Supervisors
3. Student Team / Developers
4. Users

Which is why in chapter 5 exactly these overlapping stakeholders will be used to analyze the satisfaction of the stakeholders.

Efficiency of Development Cycle

The development cycle model is not completely set by the project management approach. But in reality there is definitely a correlation between those two as we will see in chapter 5. For this evaluation it will play a role how efficient the development cycle can be when the team is working under a certain management approach.

Bottom-up evaluation using interviews

While a top-down approach using known project management success criteria might be the most structured way to inspect a project other important truths can be learned by interviewing project members. If the interviewed projects felt like something went wrong or could have been improved it will be evaluated what part caused it. These interviews will be used to examine the different project management methodologies in a bottom-up approach.

Points of Vulnerability

To find out what went wrong in the projects their points of vulnerability will be looked at. Mesly [5] suggests trying to uncover points of vulnerabilities in four key aspects of a project:

1. Plan - forecasting of project execution including schedule and external holdups
2. Process - internal mechanics / flow of all activities
3. People - collaborators and the quality of their communication / cooperation
4. Power - the lines of authority, deciding forces (management) and hierarchies/organograms of the project

These points will be used by this thesis to make the evaluation of the interviews more structured.

5. Evaluation

This chapter will evaluate different project management approaches in software development with a focus on projects in a university context using the evaluation framework set in chapter 4.

5.1. Laissez-faire

Bartol & Martin [11, p. 412] define the laissez-faire approach as following:

> *"Behavioral style of leaders who generally give the group complete freedom, Provide necessary materials, participate only to answer questions, and avoided giving feedback"*

It really depends on the situation if the laissez-faire approach works or if a more structured approach should have been used. Several papers that have shown that this approach should generally not be chosen. Chaudhry & Javed [12] show a lack of motivation under this leadership style. Anbazhagan & Kotur [13] have found that this style generally leads to the lowest productivity in teams. Skogstad et al. [14] go as far as saying that laissez-faire is not only a "zero-leadership, but a type of destructive leadership behavior".
In the special context of software development in student projects some arguments could be made for the use of laissez-faire though. One of them is that laissez-faire is often used by a management that does not have sufficient technical knowledge of a project. For student projects that work together with companies outside of the IT sector this could be the case. Usually those companies just explain their needs instead of providing technical requirements.
Still, a laissez-faire approach should only mean that the project team is not actively managed by a leadership and not that the leadership does not care about the outcome of the project. For a student project this could be the case unfortunately.

5.1.1. Top-down evaluation

This part will evaluate the laissez-faire approach from a top-down perspective.

Stakeholder Satisfaction

Project Partner It's hard to say how satisfied the external partner will be with the results of a laissez-faire approach in software development. But as one can imagine

it's not a given that the result of a project is going to be exactly as the project partner imagined if only few requirements are specified.
In a laissez-faire leadership approach they do not have to be present as much as in other approaches though. A project partner that values other projects more than the bachelor's project they are supervising would probably appreciate that.

University Supervisors The supervisors on the university side care about the educational purpose of the project. The ability to learn on their own is something that surely helps students a lot. On the other hand a project like a bachelor's project is great because it also allows the students to get in touch with more experienced people that they can gain knowledge from. In laissez-faire the contact to other personnel is expected to be not as extensive which would limit how much opportunity students have to learn from them.

Student Team / Developers For the satisfaction of the development team their expertise level most likely plays a role. Senior developers might appreciate the freedom they get. Junior developers including students could easily feel being left alone and could benefit from a more guiding leadership style.

Users How likely it is that users will be satisfied with the results coming from this approach can be answered similarly to the satisfaction of the project partner. When the team has complete freedom to develop what they think is best then they might develop something that they like but which does not consider user expectations. Therefor it is important for development teams to have short feedback cycles that keep customers in the loop to be able to steer in a direction that benefits them the most. A review of the benefits and challenges of user involvement by Kujala [15] uncovered that *"User involvement is clearly useful and it has positive effects on both system success and user satisfaction"*. Since the *laissez-faire* approach does not involve the user to a relevant degree those positive effects can not be achieved. Especially with this in mind it is most likely that users are not going to be fully satisfied with a product coming out of a *laissez-faire* project management approach.

Efficiency of Development Cycle

For most agile methodologies and frameworks more parties than the development team are needed. A Scrum team for example needs someone on the customer side that has a vision of the product and helps the team fill their product backlog with correctly prioritized *user stories*. In Scrum this role is called the *product owner*. More definitions can be found in "The Scrum Guide™" by Schwaber & Sutherland [16]. When the customer just leaves the development team on their own without providing a counterpart to fill a role like that of a *product owner* they are restricted in their choosing of a framework. With the chosen framework being linked to the development cycle at least to some degree it is questionable if the team can be as efficient as possible with this limitation

put on them.
″Small releases″ is one of the 12 practices of XP [17]. Small releases are used to have a continuous process to constantly deliver value to the customer. The intention to release often can soon fall out of focus when the team uses is under a laissez-faire approach which does not have fixed iteration like other frameworks.

5.1.2. Bottom-up evaluation

This part will evaluate the laissez-faire approach from a bottom-up perspective based on the interviews conducted.

Plan In the laissez-faire approach there seemed to be no plan. While the project partner did specify a title of the project in the beginning and maybe even a goal they did not seem to know what their requirements were for the end product. Because of this they let the student team decide on what features to implement. If they decided on the right ones is questionable.

Process The process of a software project has a lot to do with the development cycle as that is what keeps iterating and produces an output. Bartol & Martin [11, p.412] said that laissez-faire leaders generally avoid giving feedback. This was also found by the interviewed teams. It remains undecided if that was a conscious decision by the project partner or just something that went along with this behavioral style.
The team that will be talked about now had a project partner in the IT sector. So missing technical knowledge was not the issue here. But it was brought up that the team initially planned on releasing early and often and the partner then put off reviewing what the team produced for a long time. Because of this their intention soon was let go. This also negatively shaped the management framework of the team as it limited them from using certain ones. ″We do not have to think in two week iterations if our partner does not do so″ was the comment by one student out of this team.

People Laissez-faire seems to be a lot about people. The interviews brought up that supervisors in this approach seemed to care less about the outcome of the project than other supervisors. One student went as far as commenting that ″it seemed like the bachelor′s project was advertised just to have a bachelor′s project". In general it was noted by the students that they did not like when their project partner did not take at least some initiative in the progressing of the project.

Power The interviews brought up that in laissez-faire it could rather be talked about missing power. Authorities did not seem to care as much about the project outcome as in other approaches. The pace of the team seemed to be mostly set by themselves which allowed them to make it sustainable for them. That is good of course. It was also mentioned though that sometimes a little more external power that pushes something forward would have helped them succeed.

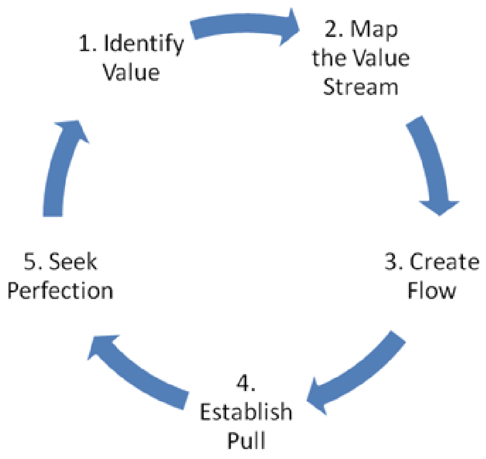


Figure 5.1.: Five-step process for lean techniques[7]

5.2. Lean Project Management with Kanban

Lean Project Management (LPM) discovered the value of lean concepts like lean construction, lean manufacturing and lean thinking and applied them to project management. E. Ries also applies the lean way of thinking to the management of a startup in ”Lean Startup” [18]. LPM has the goal of bringing value while reducing waste. Waste is defined as something that does not add value while value is defined as ”any action or process that a customer would be willing to pay for”.

The five principles of lean are explained on `lean.org`[8] as following:

1. Specify value from the standpoint of the end customer by product family.
2. Identify all the steps in the value stream for each product family, eliminating whenever possible those steps that do not create value.
3. Make the value-creating steps occur in tight sequence so the product will flow smoothly toward the customer.
4. As flow is introduced, let customers pull value from the next upstream activity.
5. As value is specified, value streams are identified, wasted steps are removed, and flow and pull are introduced, begin the process again and continue it until a state of perfection is reached in which perfect value is created with no waste.

Kanban is a framework that stems from lean production but is often used in software development as well to make it agile. Because of how many development teams use it, Kanban will be inspected more closely. Kanban follows the ”just in time” approach which means that work is pulled instead of pushed. A Kanban approach is usually aided by a virtual or physical *Kanban board* which is used to visualize the workflow in

[7]`https://www.lean.org/images/5stepslean.gif`
[8]`https://www.lean.org/WhatsLean/Principles.cfm`

which tickets move from left to right through various development stages (e.g. "To do", "In progress", "Code review", "Done").
Kanban provides added flexibility in planning what work to do next. In Kanban the development team only focuses on work that is in the column "in progress". As long as no one is already working on a ticket a ticket's scope can be changed at any time. Correctly prioritized tickets ensure that the team always starts working on tickets that result in the most value created for the user first.
In Kanban it is also possible to limit the number of tickets in each column. For example a limitation of no more than three tickets in "Code Review" could be set. While this sounds like it would hinder the development team at first it does have a supporting reason behind it. By only allowing three tickets to be in "Code Review" a developer that want's to move another ticket in this column would be forced to review someone else's work before being allowed to open his own *pull request* (PR) if the limit is reached.

5.2.1. Top-down evaluation

This part will evaluate the Kanban approach from a top-down perspective with known project management success criteria.

Stakeholder Satisfaction

Project Partner The project partner wants to profit from a partnership with a university by having a student team work on something that they can merge into their own systems. Since LPM pays close attention to always providing *value* which was defined as something that a customer would be willing to pay for the project partner should be happy since they have a special focus on building products that can go to market.
Also as there are no sprints in Kanban and the backlog can be reprioritized at any time the project partner has more flexibility. If they suddenly need a feature sooner than expected or something comes up they can simply change the order of the tickets in the backlog and rest assured that the development team will get to it next.

University Supervisors The university supervisors share the wishes and expectations of the project partner to some extend. They want the students to produce something valuable throughout the project. They are not the ones who might have to provide a *product owner* though. So losing working hours of one of their employees is not one of their concerns. University supervisors care more about the possibility for the students to learn something in the project and the output of the project to be good. As Kanban measures the success in *throughput* this aspect should be fulfilled. The question is if a Kanban project provides the student team enough opportunities to gain experience. In this aspect it probably does not matter much what framework the team uses as long as it is able to change directions fast enough when the project seems to go in a direction that does not serve an educational purpose.

Student Team / Developers The development team should feel like it does meaningful work as things that do not bring value are reduced and are probably already existing things which developers are probably not too fond of working on anyway as they most likely want to work on something new. At the same time some developers feel like they ″just assign themselves one ticket after the other and move it from left to right″. For most developers this is the dream though. They are able to focus on the coding work itself and do not spend as much time in meetings as with other methodologies like Scrum. That is due to the fact that the Kanban board is always filled and developers do not spend time to plan new sprints.
Most Kanban teams also do not estimate tickets or at least do not have to give an as precise estimation as in Scrum. In Kanban there is no need to be as precise in estimating as the team does not have to commit an amount of work to a fixed time box. While estimation is optional some teams still do it to make sure tickets do not experience *scope creep* and know when to split them up.
The fifth step of LPM ″Seek Perfection″ (Figure 5.1) is all about developing the lean spirit (″*kaizen*″) in every employee. It's not about one employee applying lean principles perfectly it's about all employees applying them as well as possible. Lean Thinking puts a focus on the transformation of the collective by causing improvements in each part of it. This mentality of accelerating change by including everybody tells the personell something good about the priorities of the company. People generally like to be included in decisions and lean methodologies improve the overall productivity. Womack, Jones & Roos first found this in their study on the future of the automobile industry [19] when evaluating the now famous Toyota Production System (TPS).

Users Users get what they want as the team has to focus on what brings value to the users. With the a lean approach and *pulling* instead of *pushing* the time until something users ask for is implemented is shorter too.

Efficiency of Development Cycle

Kanban offers continuous delivery as features are constantly put through and add to the existing system without having to be reviewed by a product owner or sitting until the end of an iteration. So a Kanban team works one ticket at a time. A Scrum team almost does so as well but actually it's rather in batches of work. However small those batches might be.
One aspect of Kanban that a team needs to keep an eye on the effort that goes into a ticket. As estimating is optional in Kanban special care needs to be taken that tasks are refined enough for developers to start working on them. Big tickets often resulted in lower motivation to start working on them in our team.
The possibility to limit work in Kanban helps to make sure that all parts of the workflow are given an equal amount of attention. It ensures that code reviews are initiated in a timely manner. This allows problems to be found quicker. And every found problem helps the team move in the direction of *kaizen* just like it helped workers in manufactur-

ing [19].
Also no matter how efficiently the developers worked their work is wasted if the acceptance of users is not high enough. "Lean Startup" [18] emphasizes that teams should always do an acceptance test after implementing a new feature. In Kanban this can be enforced by adding a new "Acceptance Test" column and putting a limit on it. After testing the acceptance of the users the team can intervene if their work seems to go in a direction that does not benefit their target group.
The same limit can be put on *Pull Requests* to shorten the time until code is reviewed.

5.2.2. Bottom-up evaluation

This part will evaluate Kanban with a bottom-up perspective based on the interviews conducted.

Plan One team using Kanban mentioned that their project partner "did not really know what they wanted when starting the bachelor's project. They did not make a plan." Over the time of the project they still managed to build a product that benefits the project partner he said. They doubted that they would have been able to do a turnaround like this with a project management approach that is not as agile as Kanban.

Process Kanban unlike other approaches does not have time boxed intervals after which one iteration ends and is reviewed. This caused one student to comment that working with Kanban felt like working on "one continuous mush". He explained that his team did not have fixed meetings in which they would look back at what they achieved in a certain time frame (e.g. the last 4 weeks) and from that make a structured plan for the next time frame. That's what made him feel the way he did.
It was also brought up by one team that they would have not been able to answer the question how much they will be able to achieve during the next month. They did like that they did not have any overhead of significant amount. But this missing overhead is also what causes the team to not be able to answer this for external stakeholders quite relevant question.

People One student commented that his development team used Kanban because "the student's schedules were not fixed enough for Scrum to work properly". He explained that "different people have different assignments outside of the bachelor's project like presentations or group assignments that need to be coordinated with other students". His group's members usually did not work a structured 9-to-5 schedule but rather were forced by assignments and spontaneous tasks to spend more time on tasks not related to the bachelor's project during one week but made up for it in the next week. He said because of this they decided on using Kanban as Scrum sprints would have been too hard for them to be planned with confidence. When taking a look at what caused this the main point is that students seem to have a not as fixed schedule as full time workers.

Power Because Kanban allows items that are not actively being worked on yet to be reprioritized and changed at any time there is no protection of a fixed duration as in other approaches. Although this should not equal granting management the possibility to introduce work with short deadlines it was brought up by a team that they felt like it was not as discouraged in Kanban. They criticized that their team sometimes had to add a newly invented feature in a short amount of time. As already said this should not have been the case even with Kanban as a framework but in reality this problem seemed to be more common in Kanban teams than in teams using other frameworks (e.g. Scrum).
These points from the interviews suggest that for external stakeholders it is easier to apply pressure on Kanban teams than on other teams. This could lead to a misuse of power or at least a not favorable situation for the students.

5.3. Iterative development with Scrum

Cusumano et al. [20] have shown that traditional phased models like the *waterfall model* are not well fitted for projects with high uncertainty and fast changing requirements. IT projects usually fall under these criteria as new technology is constantly being released and companies want to be able to adapt to it. Iterative approaches try to guarantee exactly that. Instead of having one big "planning and design" phase before starting to work on a project iterative approaches try to constantly have planning sessions that are used to steer a project in the right direction. These type of approaches have the goal of adding new features often, testing their acceptance and iterating based on the results. With this goal in mind iterative approaches are able to change directions way more often as the feedback loops is shorter and are better prepared for changing requirements from outside.
Scrum counts to the most used of the iterative approaches. This claim was supported by the teams interviewed for this thesis. Therefor this thesis will take a closer look at the Scrum framework.

Scrum is an agile framework with various processes and techniques that can be put to work by teams to "address complex adaptive problems, while productively and creatively delivering products of the highest possible value" [16]. One of Scrum's events are sprints which are utilized in an attempt to including other things better estimate a team's progress in the future. Sprints are defined in [16] as

> *"a time-box of one month or less during which a 'Done', usable, and potentially releasable product Increment is created. Sprints best have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint."*

What features the development team works on is defined in the *product backlog* by the *product owner*.

One major rule of Scrum is that the *product owner* ought to refrain from changing the requirements and prioritization of features during a running iteration. In that aspect the development team experiences a little protection.

5.3.1. Top-down evaluation

This part will evaluate the Scrum approach from a top-down perspective with known project management success criteria

Stakeholder Satisfaction

Project Partner The project partner has to provide a *product owner* to the development team to act as a counterpart. Of course that means freeing a part of the time of a person that could otherwise work on something else like a project inside the company. Since the company gets to put a team of students (usually consisting of 6-8 people) on something they will profit from as well without having to monetarily compensate them this should not be an issue though. This outweighs the added cost of having to provide a *product owner* to the team. In our project "BAAM" the *product owner's* time was not occupied for more than two hours on average per week.
Project partner supervisors want to see the team's progress to be able to evaluate the project's success. In Scrum there are scheduled meetings for this in the form of *sprint reviews*. In other project management approaches they would have to schedule these meetings and tell the team to prepare to demonstrate their progress. In *sprint reviews* this is firmly integrated into the whole process and with sprint planning tools like JIRA the development team can easily recount what happened since the last meeting took place. Scrum also has the goal of always having functioning code. Considering this a demonstration should always be possible at the sprint end.
Scrum allows the *product owner* to reprioritize tickets at any time as long as they are not part of a running sprint yet. This provides them with added flexibility compared to more traditional approaches like the *waterfall model* where every requirement has to be specified before starting the project. The limitation Scrum puts on the *product owner* is that they have to respect the protection of a running sprint. That means that they will have to wait until the start of a new iteration (usually lasting two weeks) to have the development team work on different features. This protection of a running sprint's scope was introduced to let the development team work on tasks which requirements will not change while they are working on it to prevent *context switches*.
A Scrum team is able to change directions after every iteration though. The timebox for that was suggested by Sutherland [16] as one month or less. In the interviews conducted for this paper it was found that most of the interviewed teams had sprints of two weeks duration so even shorter than one month.

University Supervisors As already mentioned in the evaluation of stakeholder satisfaction in Kanban (see subsection 5.2.1) university supervisors pay special attention to the

question if the project fulfills its academic purpose.
Like Kanban the Scrum framework does not pay more attention to this part than any other framework. But the agility of Scrum should enable the university supervisors to intervene soon and often enough if the project seems to go in a direction that does not serve educationally. "The Scrum Guide™" [16] lists an event called *retrospective* as part of the framework. This event is used to reflect on how the last sprint went. *Retrospectives* could also be used to evaluate the educational usefulness of the project.

Student Team / Developers The developers are protected from changing requirements during a running sprint so depending on what they agreed to probably at least for two weeks at a time. That provides them with a little bit of safety from external stakeholders. This added barrier between the development team and external demands is something that they will appreciate. The goal of the team during the sprint is that no work committed to remains unfinished at the end of it. So every sprint end acts as a mini deadline for the student team. This deadline can both hurt and help the development process. For one it pushes the team to work a little harder towards the end of it. Working harder should not be confused with working overtime. When teams are forced to work overtime they are not working at a sustainable pace and that hurts the overall progress which is why "sustainable pace" is one of the 12 practices of XP [17]. It should only mean that the team pays a little more attention to staying focused on the last mile. If it would force them to work overtime then they overestimated their velocity and can adjust it for the next iteration.
When evaluating the satisfaction of the development team with Scrum another thing comes to mind. Scrum has a higher overhead for the team compared to an approach like Kanban. Sprints need to be planned and tickets estimated. A case study by Taibi et al. [21] found that the communication effort is higher in a Scrum process than in a Scrum in combination with Kanban ("Scrumban") process. They did not test if a pure Kanban process would have resulted in even less communication effort but still their findings illustrate a point. When asked developers usually respond that they would like to spend less time on project management overhead and more time on coding (more on this in the bottom-up evaluation in subsection 5.3.2).

Users For users it should not make much of a difference if the development team works in a Scrum or Kanban setting except that they are affected by the efficiency of the development cycle because of how fast the product reaches them. So no further analysis will be done here. They get the same advantages of short iterations and feedback cycles and therefor the points brought up in the satisfaction of users in Kanban (section 5.2.1) apply here as well.

Efficiency of Development Cycle

This part takes a look the efficiency of the development cycle of a team that uses Scrum. The Scrum framework aims to iteratively add new features to a system or

product. Through this type of development the "small releases" principle from the 12 XP principles [17] is enforced.
One aspect is that Scrum's sprint reviews can hurry the team into making errors or can lure them into not putting as much time in reviewing their code. This of course increases the probability of introducing bugs into the system. Kononenko et al. linked the reviewer workload to the quality of the code review process [22]. Especially when higher level executives are present in *sprint reviews* the temptation for the team is bigger to ignore last concerns. It should be mentioned though that this reaction of the team can be prevented by setting a good working environment. If supervisors and managers communicate clearly that quality and therefor properly reviewed code is more important to them than meeting those mini deadlines the team can represent their work honestly. The protection of a sprint's scope saves *context switches* and that saves the students time that they can in turn spend on the tasks on hand. Overall the development cycle in Scrum seems to be efficient.

5.3.2. Bottom-up evaluation

This part will evaluate the Scrum approach from a bottom-up perspective based on the interviews conducted.

Plan The team of "BAAM" used the Scrum framework. Over the course of the project it became evident that the initially defined roadmap was too optimistically planned. When planning the effort of certain features was underestimated. Additionally the team experienced some external blockers that they had to wait on. This was not considered enough when defining the schedule before project start. By choosing an agile approach luckily the team in joint effort with the *product owner* was still able to implement the most important features before the end of the project by changing the prioritization or scope of some features. This real-world scenario illustrates that even though the initial plan was not perfect the project team was still able to produce a product that in the end satisfied the project partner. In a project management approach like the *waterfall model* the initial planning would have required more effort and would probably still have had flaws as Cusumano et al. [20] have shown.

Process One thing that came through in the interviews was that developers do not appreciate when a framework has a lot[9] of overhead for them. At least some of the developers of every Scrum team felt like the overhead of this approach was too much. They mentioned that they would have had more time to implement code if they did not have such an amount of overhead because of estimating tickets, planning sprints and more. It can not be argued against the fact that they spent more time on tasks that are not directly related to implementing code. The question is though how efficient and effective the overall process was. Just because developers spent less time on writing

[9]Although they did not specify what exactly "a lot" means for them

code it does not mean that the team brought less value to the product in the same time. The developers felt like Scrum was an effective framework for projects with changing requirements.

People In section 4.1 it was already brought up that students generally have less experience than more seasoned developers working in industrial projects. Estimating how much effort will go into a certain feature would be a complicated task for most people. For not as experienced students it is probably even harder. This hypothesis was confirmed by several students that were interviewed. This puts a vulnerability on the software project by the people working on it.

Power The Scrum approach asks for a *product owner*. They are the ones that fill the product backlog and thereby decide what features are to be implemented. The *product owner* is not to be confused with a *project manager*. They are not above the development team in terms of hierarchy. To put it correctly they are part of the team and are simply responsible for a different part of the project. Instead of being responsible for the technical side of the project they are responsible for the product side of the project by steering the product in the right direction. But although they are not above the project team and instead are an included part of it they tell the team what to work on. Yet they do this from inside of the team and respectively have people they have a responsibility to. The development team or in this case to be more precise the students have the *product owner* shielding them from the requirements of external stakeholders. "The role of the product owner is vital for our team to effectively manage the expectations of our external stakeholders." was said by the team of the industrial project. Since student teams do not seem to differ from industrial teams in this aspect this statement is just as true for student projects. If Scrum is adopted correctly the lines of power should be in favor of the team.

5.3.3. Further insights from project "BAAM"

Our project "Behavioral Authentication Access Management" employed Scrum. From using it in a practical setting we were able to gain valuable experience on how well it works in the context of a bachelor's project or a student project in general for that matter. The findings of using Scrum will be split in different categories in the part to come.

Mistakes in sprint planning

This part will take a look at some of the mistakes we made when planning sprints for our iterative approach using the Scrum framework.
Figure 5.2 shows the work (in *story points*) done over time during sprint 8 of project "BAAM". Explicitly it shows the remaining work in contrast to the ideal trend. Before the start of the sprint the development team committed to 54 *story points* and increased

it to 59 *story points* (SP) right after starting the sprint by adding another ticket. At the end of the sprint 6 SP remained. Compared to the gray guideline one can see that the development team stayed relatively true to what they committed to do over the course of the sprint. It should be noted that in this project the remaining work for a ticket was not updated after finishing a day's work or so. That explains that the burndown looks more like a "staircase" rather than a straight line. *Story points* only count as done when the corresponding ticket is in the *done* column of the JIRA board. Partial work done is not reflected in the burndown charts. With that in mind it can be concluded that the planning on the sprint went really well in terms of estimating tickets and deciding on a workload to which to commit to for this sprint.

Another sprint burndown chart is given in Figure 5.3. This time the team committed to 144 SP. Over the course of the sprint the remaining work continuously goes down and progress is made. Yet, at the end of the sprint a major part of the work committed to still remains unresolved. As for actual numbers: 37 SP were completed in this sprint, 107 remained at the end of the sprint. So what went wrong?
A vital part of Scrum is the protection of the Scrum team during a running sprint. Lets focus on the sprint shown in Figure 5.3 again. Was this protection adhered to? When taking a look at the sprint report of this sprint (Figure A.1,Figure A.2) this question can be answered positively. Every issue that was added after the start time is marked with an asterisk. Actually of the completed issues several are marked this way. But the issue type indicates that they are all bugs and those were found by the team and fixed right away. The *product owner* did not have anything to do with it. The student team chose to handle bugs this way on their own. Issues that were added during the sprint and are not bugs were something the team added to split up existing tasks or something they added to the sprints to be able to develop other features more efficiently (i.e. "BAAM-431: Adjust ESLint Settings"). The graph also supports this as except for a small change on May 16th the remaining work does not increase. So adding additional work after the sprint had already started was not the issue here.
When comparing the workload the team committed to in this sprint with the velocity of other sprints (see Figure 5.4) one can see that the planned workload was higher than usual for this sprint. Especially compared to sprint 8 (Figure 5.2) which was just identified as a well estimated iteration. It does have to be noted that in sprint 8 the team was still working 20 hours a week instead of full time (40 hours) like later. And also one developer was not present during that sprint. It seems like the smaller size of the sprint made it easier to estimate it.
To conclude that all sprints should be smaller is nevertheless a little premature as the overhead of sprints is expected to increase too. After sprint-13 107 SP remained. With this many *story points* remaining the team can feel like they did not manage to do enough work. The produced output would have been the same if fewer *story points* were added to the sprint. But the remaining work after the sprint would have been closer to zero. That would have allowed the development team to feel more satisfied with this sprint and go into the new sprint with more confidence.

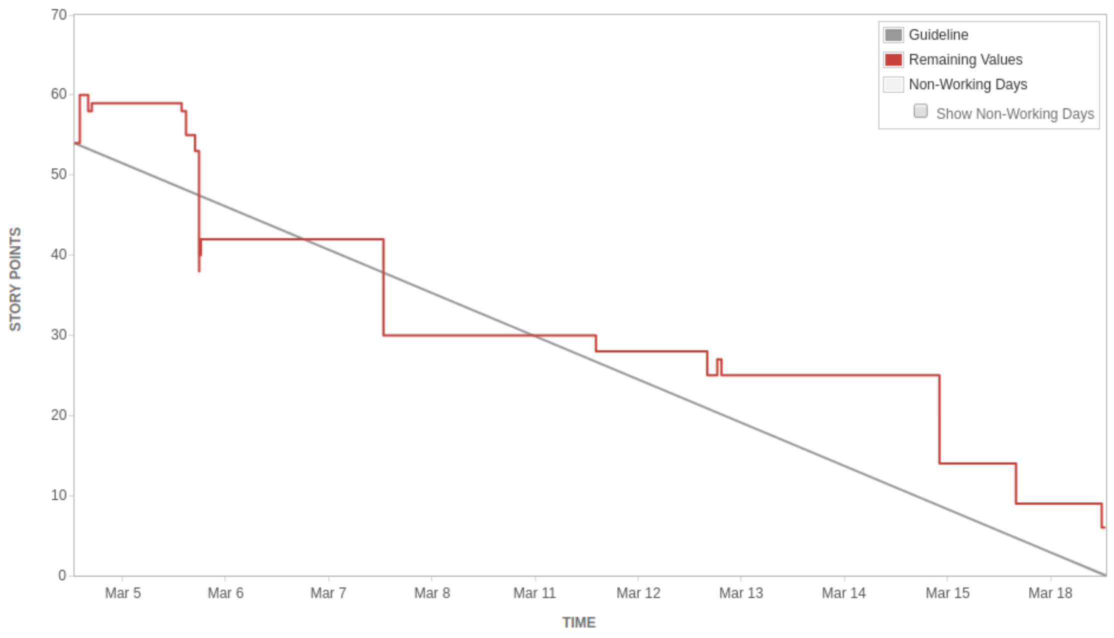


Figure 5.2.: Burndown chart of Scrum sprint 8 of "BAAM"

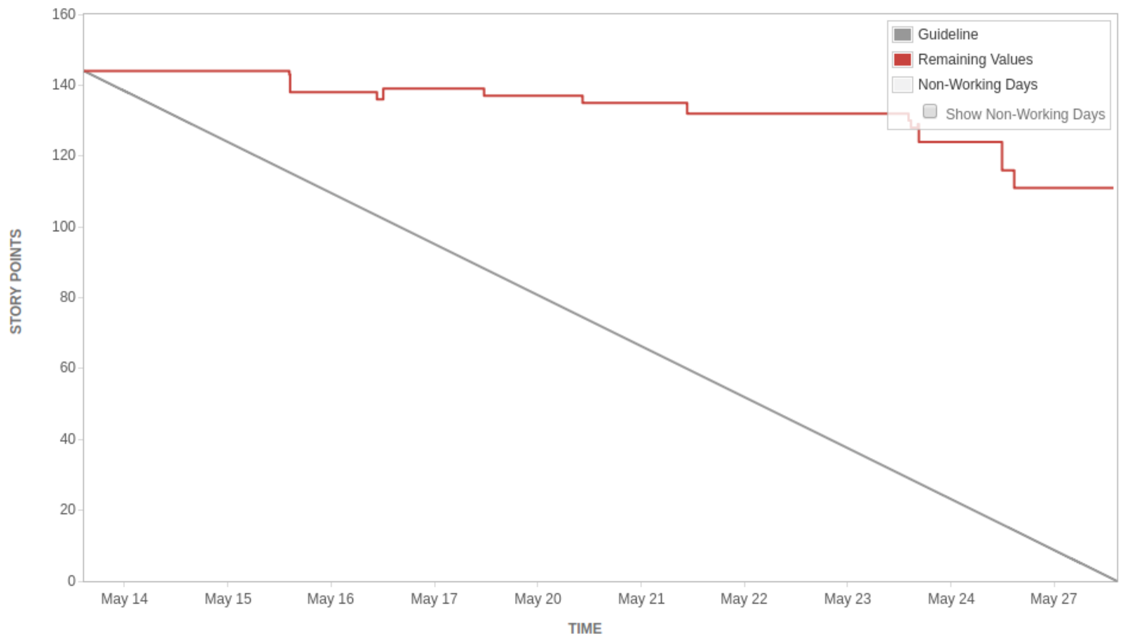


Figure 5.3.: Burndown chart of Scrum sprint 13 of "BAAM"

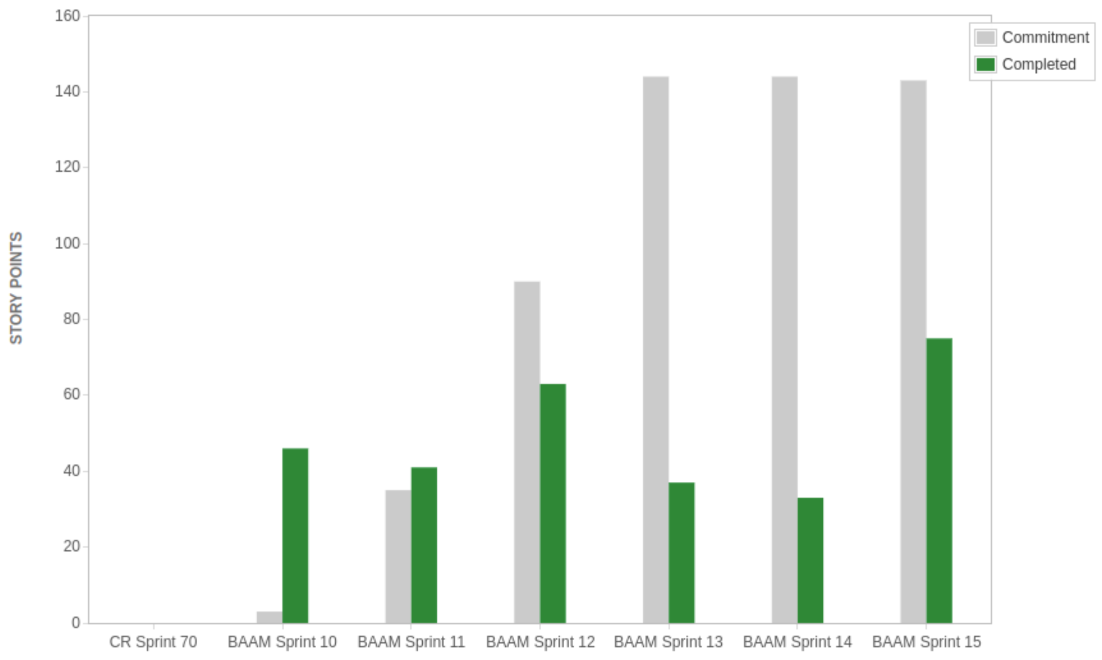


Figure 5.4.: Velocities of "BAAM" sprints 10-15

Mistakes in development cycle

At first when working part time on our project code did not have to wait long on being reviewed. After transitioning into full time we noticed that code was not reviewed as frequently. Our motivation to do so had gone done now that people were able to produce more code in a day. We addressed this issue by only allowing three *Pull Requests* (PR) max to be open at the same time. This forced developers to review someone else's PR before opening a new one. Without planning on it our Scrum team adopted one feature of Kanban by doing so.

5.4. Comparison

When looking at the expected satisfaction of the stakeholders it can be noticed that the satisfaction of users and university supervisors seems to be higher in Kanban and Scrum than in the laissez-faire approach. It does not seem to differ much between the first two though.
The expected satisfaction of students and project partners does change when using a different project management approach on the other hand. Scrum seemed to be better at laying out what will happen in the near future and therefor resulted in higher expected satisfaction for project partners and university supervisors in that aspect.

The student satisfaction was the lowest in the laissez-faire approach which is why this approach can not be recommended for student projects.

Students especially appreciated the reduced project management overhead in Kanban and it saved them from having to estimate what they will get done in the next iteration. It also seemed to fit their sometimes changing university schedules outside of the bachelor's project very well.
It can be seen that both Kanban and Scrum bring their own advantages and disadvantages. The interviews suggested that the students also noticed this and adjusted their project management approach to include certain practices of other approaches if it improved their approach. Especially Scrum teams seemed to often introduce Kanban practices over time and shifts to "Scrumban" were often seen.
One student's team started with all Scrum practices and gradually removed the practices that they defined as too much overhead for too little benefit for them. Nevertheless he mentioned that he would still start a new project with all practices in place at first. The not beneficial practices would soon be highlighted again he said. This approach worked very well for them and is something that could be tested when starting a project.
In general frameworks and methodologies should be seen as what they are: tools to aid the project team. If they are not helping a team as much as they thought initially then the tools should be removed or adjusted to fit the specifics of the team. A project by definition is something that is different to past projects. The usefulness of certain tools should be just as critically evaluated by the team as their development tools. After all they influence the project's outcome equally as much.

Barash [23] when analyzing the *"Use of Agile with XP and Kanban Methodologies in the Same Project"* concluded "Development and reporting work is best suited to prescriptive flow including timebox methods prescribed by Agile Scrum. Infrastructure work with interactions to internal departments and external suppliers is best completed utilizing the natural adaptive flow of Kanban.". So as it can be seen it is also worth thinking about a mixture of multiple methods. Our project "BAAM" confirmed that the communication effort for infrastructure related inquiries was higher than for other inquiries. This suggests that the main part of what the students will be working on does play a role in the choosing of a framework.

The main question now is if LPM or an iterative approach would be better suited.
All in all the interviews did suggest that Kanban was a little more liked by the students as it accompanied the fact that the students did not have much experience in estimating efforts yet and did not want to have too much overhead. Scrum seemed to be a little better for the project partner and the university supervisors as it has fixed meetings in which progress can be measured. In the context of student projects the question if it benefited the students is of special importance though. Considering the usual goals of a student project they might be the most significant stakeholder of them all. After careful evaluation Kanban seems to lead to more satisfaction for the students and since in many aspects like the efficiency of the development cycle Scrum and Kanban did not seem to make much of a difference it is fair to make this the deciding point.

If the team does choose to employ a lean project management approach with Kanban then special care must be taken that meetings that evaluate how well the project is going still take place. If those meetings do not take place the risk exists that the students will still not be satisfied with the course of the project.

5.5. Limitations

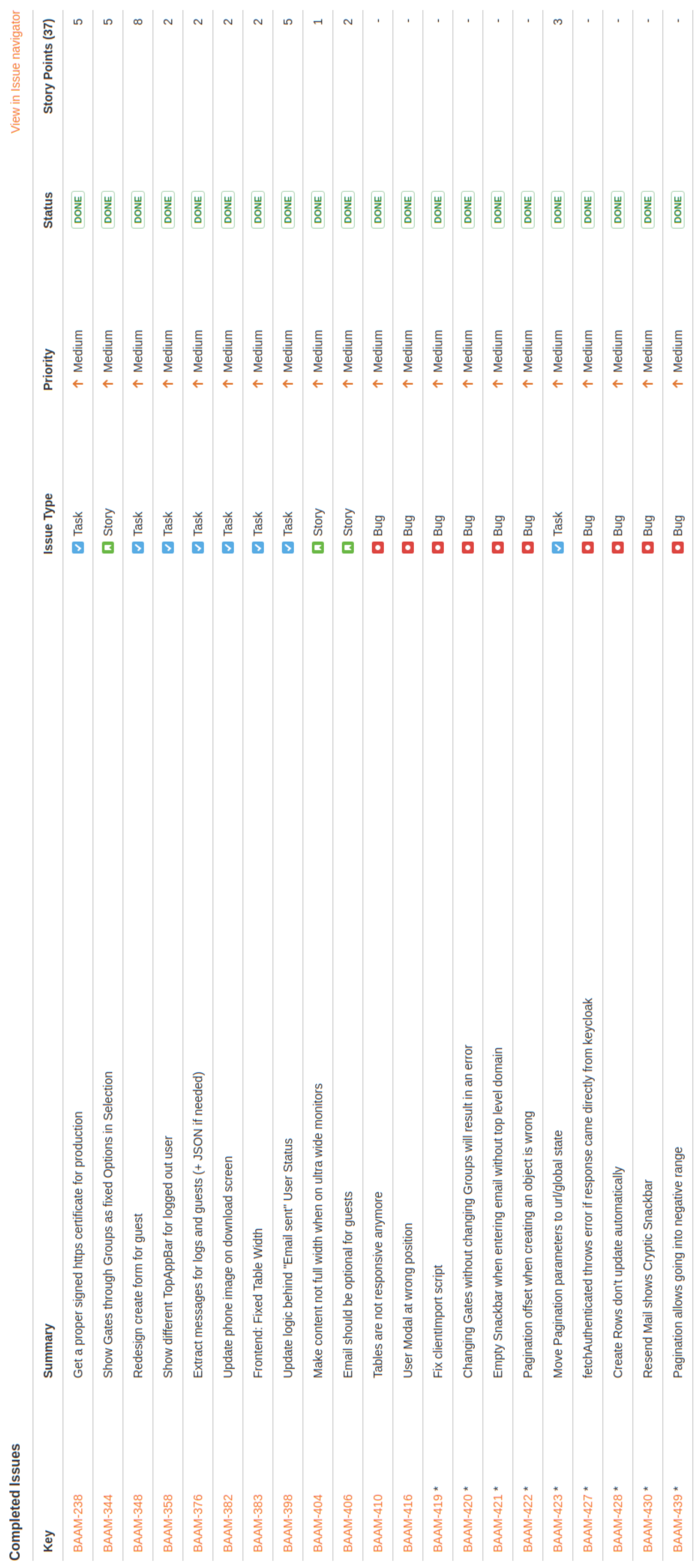
It is important to point out some things that might limit the validity of this thesis. This thesis tried to not only evaluate the ″BAAM″ project to not have too much of a bias. Still it only included six projects total into the evaluation one of which was not a student project but used more as a reference. The sample size warrants a little caution about projecting the things said in six projects to all student projects. The projects interviewed for this thesis were also all part of the same university. One reason for not being able to benefit from more project's experiences is that for the focus of this thesis only projects that actively worked in teams were fitting. Other projects were more research oriented instead of building a product that is meant to go to market. Those projects usually had every member working on their own.
To accommodate the small sample size this thesis chose to view the interviewed projects more as impressions of what students felt like and put it in words instead of in numbers. Furthermore even though more projects than just ″BAAM″ were interviewed and this thesis tried to be as generically applicable as possible the author was still part of one bachelor's project himself (″BAAM″) which used Scrum and it must be assumed that the author could not completely refrain from being a little biased.

6. Conclusion and future work

The goal of our project "Behavioral Authentication Access Management" was to build an access management platform that accommodates the special challenges of behavioral authentication in physical access management. Real-world experiences from this project as well as other projects were used to evaluate different project management approaches for software development in the special context of student projects.
Before evaluating the different approaches an evaluation framework was set that is based on a top-down as well as a bottom-up approach. A differentiation between student and industrial projects was also provided to weight the findings of the evaluation based on their significance for student projects before coming to a comparison.
The comparison of the different approaches found that in general the *laissez-faire* approach seemed to perform worse than a *lean project management* approach with *Kanban* and an *iterative* approach with *Scrum*.
Between the last two approaches certain differences in advantages and disadvantages were shown. The choosing between these two mostly comes down to the preferences of the project team as in many top-down findings both approaches performed well in some aspects while losing in others. Kanban was a little more favored by this thesis though because of the realistic findings of the bottom-up evaluation. The two main reasons for it were the reduced overhead for students and the better fit for their university schedules outside of the project.
The bottom-up findings also showed that most teams adjusted the project management frameworks to fit their specific needs and that it served them well. Due do that this thesis also suggested a critical review of the tools used and to adjust them if needed.

In the future more statistic data could be gathered that could then be used to evaluate different project management approaches on their output produced. Due to the small sample size available and the fact that different projects also worked on different challenges a direct comparison based on numbers was not possible for this work. Future work could compare the code produced and rank it based on metrics like quantity and quality. It could then be concluded if the project management approach in student software projects does actually significantly influence the produced output or if other factors like the experience of students play a more important role.
Future work could also focus on projects that differ from the projects presented in this work. For instance this work did not pay much attention to research projects in IT which often still have a significant share of software development but also include other tasks. Evaluations on research projects with a development part would most likely need to set a different evaluation framework.

A. Appendix

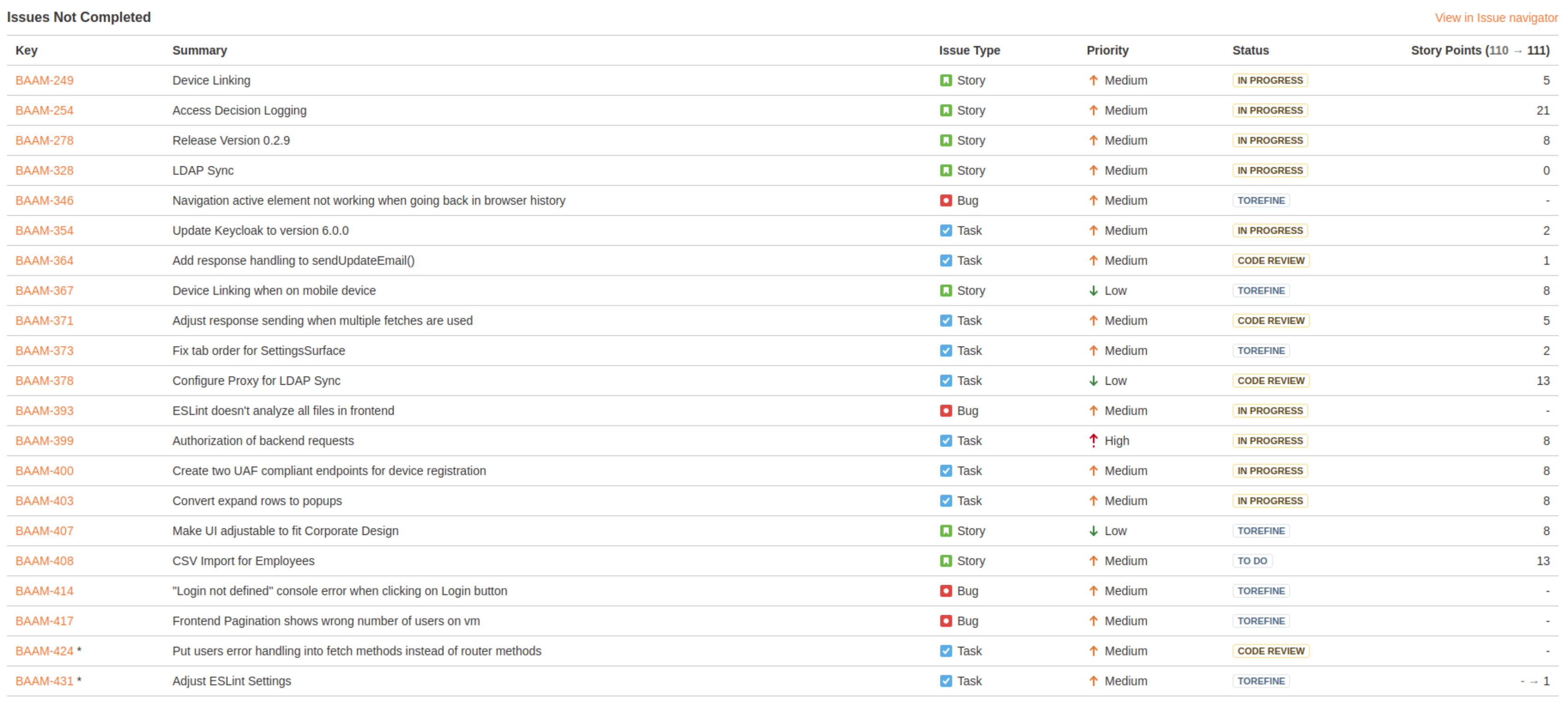


Completed Issues

View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (37)
BAAM-238	Get a proper signed https certificate for production	Task	↑ Medium	DONE	5
BAAM-344	Show Gates through Groups as fixed Options in Selection	Story	↑ Medium	DONE	5
BAAM-348	Redesign create form for guest	Task	↑ Medium	DONE	8
BAAM-358	Show different TopAppBar for logged out user	Task	↑ Medium	DONE	2
BAAM-376	Extract messages for logs and guests (+ JSON if needed)	Task	↑ Medium	DONE	2
BAAM-382	Update phone image on download screen	Task	↑ Medium	DONE	2
BAAM-383	Frontend: Fixed Table Width	Task	↑ Medium	DONE	2
BAAM-398	Update logic behind "Email sent" User Status	Task	↑ Medium	DONE	5
BAAM-404	Make content not full width when on ultra wide monitors	Story	↑ Medium	DONE	1
BAAM-406	Email should be optional for guests	Story	↑ Medium	DONE	2
BAAM-410	Tables are not responsive anymore	Bug	↑ Medium	DONE	-
BAAM-416	User Modal at wrong position	Bug	↑ Medium	DONE	-
BAAM-419 *	Fix clientImport script	Bug	↑ Medium	DONE	-
BAAM-420 *	Changing Gates without changing Groups will result in an error	Bug	↑ Medium	DONE	-
BAAM-421 *	Empty Snackbar when entering email without top level domain	Bug	↑ Medium	DONE	-
BAAM-422 *	Pagination offset when creating an object is wrong	Bug	↑ Medium	DONE	-
BAAM-423 *	Move Pagination parameters to url/global state	Task	↑ Medium	DONE	3
BAAM-427 *	fetchAuthenticated throws error if response came directly from keycloak	Bug	↑ Medium	DONE	-
BAAM-428 *	Create Rows don't update automatically	Bug	↑ Medium	DONE	-
BAAM-430 *	Resend Mail shows Cryptic Snackbar	Bug	↑ Medium	DONE	-
BAAM-439 *	Pagination allows going into negative range	Bug	↑ Medium	DONE	-

Figure A.1.: Completed issues of scrum sprint 8 of "BAAM"



Issues Not Completed

View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (110 → 111)
BAAM-249	Device Linking	Story	↑ Medium	IN PROGRESS	5
BAAM-254	Access Decision Logging	Story	↑ Medium	IN PROGRESS	21
BAAM-278	Release Version 0.2.9	Story	↑ Medium	IN PROGRESS	8
BAAM-328	LDAP Sync	Story	↑ Medium	IN PROGRESS	0
BAAM-346	Navigation active element not working when going back in browser history	Bug	↑ Medium	TOREFINE	-
BAAM-354	Update Keycloak to version 6.0.0	Task	↑ Medium	IN PROGRESS	2
BAAM-364	Add response handling to sendUpdateEmail()	Task	↑ Medium	CODE REVIEW	1
BAAM-367	Device Linking when on mobile device	Story	↓ Low	TOREFINE	8
BAAM-371	Adjust response sending when multiple fetches are used	Task	↑ Medium	CODE REVIEW	5
BAAM-373	Fix tab order for SettingsSurface	Task	↑ Medium	TOREFINE	2
BAAM-378	Configure Proxy for LDAP Sync	Task	↓ Low	CODE REVIEW	13
BAAM-393	ESLint doesn't analyze all files in frontend	Bug	↑ Medium	IN PROGRESS	-
BAAM-399	Authorization of backend requests	Task	↑ High	IN PROGRESS	8
BAAM-400	Create two UAF compliant endpoints for device registration	Task	↑ Medium	IN PROGRESS	8
BAAM-403	Convert expand rows to popups	Task	↑ Medium	IN PROGRESS	8
BAAM-407	Make UI adjustable to fit Corporate Design	Story	↓ Low	TOREFINE	8
BAAM-408	CSV Import for Employees	Story	↑ Medium	TO DO	13
BAAM-414	"Login not defined" console error when clicking on Login button	Bug	↑ Medium	TOREFINE	-
BAAM-417	Frontend Pagination shows wrong number of users on vm	Bug	↑ Medium	TOREFINE	-
BAAM-424 *	Put users error handling into fetch methods instead of router methods	Task	↑ Medium	CODE REVIEW	-
BAAM-431 *	Adjust ESLint Settings	Task	↑ Medium	TOREFINE	- → 1

Figure A.2.: Remaining issues of scrum sprint 13 of "BAAM"

Bibliography

[1] Benjamin S Bloom and others. „Taxonomy of educational objectives. Vol. 1: Cognitive domain“. In: *New York: McKay* (1956), pages 20–24.

[2] *What is Agile Software Development?* Agile Alliance. June 29, 2015. URL: https://www.agilealliance.org/agile101/ (visited on July 14, 2019).

[3] A.M. Davis, E.H. Bersoff, and E.R. Comer. „A strategy for comparing alternative software development life cycle models“. In: *IEEE Transactions on Software Engineering* 14.10 (Oct. 1988). fits really well; I can use the points he uses to compare other management approaches., pages 1453–1461. ISSN: 0098-5589. DOI: 10.1109/32.6190. URL: http://ieeexplore.ieee.org/document/6190/ (visited on June 10, 2019).

[4] Amadou Diallo and Denis Thuillier. „The success dimensions of international development projects: the perceptions of African project coordinators“. In: *International Journal of Project Management* 22.1 (Jan. 2004), pages 19–31. ISSN: 02637863. DOI: 10.1016/S0263-7863(03)00008-5. URL: https://linkinghub.elsevier.com/retrieve/pii/S0263786303000085 (visited on June 11, 2019).

[5] Olivier Mesly. *Project Feasibility: Tools for Uncovering Points of Vulnerability*. Google-Books-ID: K4wmDwAAQBAJ. CRC Press, June 1, 2017. 442 pages. ISBN: 978-1-315-29523-7.

[6] O.P. Brereton, S. Lees, R. Bedson, C. Boldyreff, S. Drummond, P. Layzell, L. Macaulay, and R. Young. „Student collaboration across universities: a case study in software engineering“. In: *Thirteenth Conference on Software Engineering Education and Training*. Thirteenth Conference on Software Engineering Education and Training. Austin, TX, USA: IEEE Comput. Soc, 2000, pages 76–86. ISBN: 978-0-7695-0421-6. DOI: 10.1109/CSEE.2000.827025. URL: http://ieeexplore.ieee.org/document/827025/ (visited on July 17, 2019).

[7] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. 2001. ISBN: 1-880410-23-0.

[8] Michael Bloch, Sven Blumberg, and Jürgen Laartz. *Delivering large-scale IT projects on time, on budget, and on value | McKinsey*. URL: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value (visited on June 30, 2019).

[9] *Software developers, applications & systems software | Data USA*. URL: https://datausa.io/profile/soc/15113X/ (visited on June 30, 2019).

[10] Dale L. Goodhue. „Understanding User Evaluations of Information Systems". In: *Management Science* 41.12 (Dec. 1995), pages 1827–1844. ISSN: 0025-1909, 1526-5501. DOI: 10.1287/mnsc.41.12.1827. URL: http://pubsonline.informs.org/doi/abs/10.1287/mnsc.41.12.1827 (visited on July 8, 2019).

[11] KM Bartol and DC Martin. *Management: International Edition*. Boston: Irwin, McGraw-Hill, 1998.

[12] Husnain Javed Abdul Qayyum Chaudhry. „Impact of Transactional and Laissez Faire Leadership Style on Motivation". In: *International Journal of Business and Social Science* 3.7 (Apr. 2012). URL: http://www.ijbssnet.com/journals/Vol_3_No_7_April_2012/28.pdf.

[13] Kotur BR Anbazhagan S. „Worker Productivity, Leadership Style Relationship". In: *IOSR Journal of Business and Management* 16.8 (Aug. 2014), pages 62–70. DOI: 10.9790/487x-16846270. URL: http://iosrjournals.org/iosr-jbm/papers/Vol16-issue8/Version-4/H016846270.pdf.

[14] Anders Skogstad, Ståle Einarsen, Torbjørn Torsheim, Merethe Schanke Aasland, and Hilde Hetland. „The Destructiveness of Laissez-Faire Leadership Behavior". In: *Journal of occupational health psychology* 12 (Feb. 2007), pages 80–92. DOI: 10.1037/1076-8998.12.1.80.

[15] Sari Kujala. „User involvement: A review of the benefits and challenges". In: *Behaviour & Information Technology* 22.1 (Jan. 2003), pages 1–16. ISSN: 0144-929X, 1362-3001. DOI: 10.1080/01449290301782. URL: http://www.tandfonline.com/doi/abs/10.1080/01449290301782 (visited on July 17, 2019).

[16] Jeff Sutherland and Ken Schwaber. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game.(2016)*. 2016. URL: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf.

[17] Cynthia Andres Kent Beck. *Extreme Programming Explained: Embrace Change, Second Edition*. 2nd edition. 2012.

[18] Eric Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books, 2011.

[19] James P. Womack, James P. Womack, Daniel T. Jones, Daniel Roos, and Massachusetts Institute of Technology. *Machine that Changed the World*. Simon and Schuster, 1990. ISBN: 978-0-89256-350-0.

[20] Michael A Cusumano and Stanley A Smith. „Beyond the waterfall: Software development at Microsoft". In: (1995).

[21] Davide Taibi, Valentina Lenarduzzi, Muhammad Ovais Ahmad, and Kari Liukkunen. „Comparing Communication Effort within the Scrum, Scrum with Kanban, XP, and Banana Development Processes". In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE'17*. the 21st International Conference. Karlskrona, Sweden: ACM Press, 2017, pages 258–263. ISBN: 978-1-4503-4804-1. DOI: 10.1145/3084226.3084270.

URL: `http://dl.acm.org/citation.cfm?doid=3084226.3084270` (visited on July 15, 2019).

[22] Oleksii Kononenko, Olga Baysal, Latifa Guerrouj, Yaxin Cao, and Michael W. Godfrey. „Investigating code review quality: Do people and participation matter?“ In: *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME). Bremen, Germany: IEEE, Sept. 2015, pages 111–120. ISBN: 978-1-4673-7532-0. DOI: `10.1109/ICSM.2015.7332457`. URL: `http://ieeexplore.ieee.org/document/7332457/` (visited on July 28, 2019).

[23] Ira Barash. „Use of agile with XP and Kanban methodologies in the same project“. In: *PM World J* 2.2 (2013), pages 1–11. URL: `https://pmworldlibrary.net/wp-content/uploads/2013/10/pmwj15-oct2013-barash-use-of-agile-xp-kanban-same-project-SecondEdition.pdf`.

Eidesstattliche Erklärung

Hiermit versichere ich, dass meine Bachelor's Thesis „Evaluation of Different Project Management Approaches in Software Projects in a University Context" („Evaluierung von verschiedenen Projektmanagementansätzen in studentischen Softwareprojekten") selbständig verfasst wurde und dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt wurden. Diese Aussage trifft auch für alle Implementierungen und Dokumentationen im Rahmen dieses Projektes zu.

Potsdam, den 31. Juli 2019,

(Max Plaga)