Predictability Improvement in Agile Team Planning

Saimon Marouki

Spring 2017
Master’s thesis, 30 ECTS
Civilingenjör Teknisk Datavetenskap
Department of Computer Science, Umeå University
Examiner: Henrik Björklund
Supervisors: Ola Ringdahl, Per Nordlinder
Abstract

Over the past decade, the Agile framework has revolutionized the software industry where development teams and organizations have radically changed in how they are structured and operate. The development process which is the core component of the Agile framework, along with many other activities such as project management, team organization, stakeholder involvement and release management have essentially changed. The trend has shifted from the traditional project management methodologies which are deemed rigid and are known to cause a considerable number of issues to the lightweight and flexible methodologies that fall under the Agile framework. The shift has overcome major hurdles and issues that were experienced in the past, but smaller problems have arisen due to the dynamic nature and flexibleness of the agile software development process. Software development teams are often having difficulties making accurate forecasts for their sprints. The consequences have ranged from budget and release planning deviations to below expected product quality.

In this thesis, the aim was to investigate whether a tool that records progress and makes simple sprint forecasts would alleviate these issues and how the tool could be improved. Also, an in-depth theoretical study was conducted regarding the factors that affect the forecasts for a sprint and team progress.

The project was carried out by developing the tool as two subprojects, the back-end and the front-end, in collaboration with another student. The back-end serves the front-end which in this case is the web client, through an API. The tool was released to a test group that utilized it in their main work during an extensive amount of time. The group was then interviewed and the results were used in conjunction with the theoretical studies to propose improvements to the tool.

Results show that the tool does improve forecast accuracy to the extent that it paves way for the team in allowing it to track progress and determine their capacity. The in-depth studies show that there are several factors that are based on historical and present data that should be regarded. The factors are mainly of technical and social character.
Contents

1 Introduction ............................................................................................................. 1
  1.1 Objectives ............................................................................................................. 1
  1.2 Background ......................................................................................................... 1
    1.2.1 Agile framework .............................................................................................. 1
    1.2.2 Scrum methodology ......................................................................................... 4
    1.2.3 Roles in a scrum project ................................................................................. 5
    1.2.4 Scrum sprint .................................................................................................... 5
    1.2.5 Scrum artifacts .................................................................................................. 7
    1.2.6 User stories ...................................................................................................... 10
    1.2.7 Estimation techniques ..................................................................................... 11
    1.2.8 Sprint forecasting ............................................................................................. 13
  1.3 Aim and purpose .................................................................................................. 14

2 Method .......................................................................................................................... 16
  2.1 Project outline ..................................................................................................... 16
  2.2 Survey .................................................................................................................. 16
  2.3 Tool development .................................................................................................. 16
  2.4 Evaluation ............................................................................................................. 17
  2.5 Research ............................................................................................................... 17
  2.6 User story mapping ............................................................................................... 17

3 Results .......................................................................................................................... 19
  3.1 Survey .................................................................................................................. 19
  3.2 Tool architecture ................................................................................................. 22
  3.3 Tool evaluation ..................................................................................................... 27
  3.4 In-depth study ....................................................................................................... 27

4 Discussion .................................................................................................................... 34
  4.1 Survey review ...................................................................................................... 34
  4.2 Tool enhancements ............................................................................................... 35
4.3 Threats to validity ............................................................. 37
4.4 Future work........................................................................ 37
5 Bibliography........................................................................... 39
6 Appendix ................................................................................. 42
6.1 Survey................................................................................ 42
6.2 Interview session.............................................................. 44
1 Introduction

The software industry is increasingly making use of the Agile software development framework [1] in which Scrum has become the most adopted framework [2] in the industry. Software development teams are switching from rigid and heavyweight traditional processes to more flexible and lightweight ones. The increased transparency and the shifting of the decision power and control to the teams has come at a cost. Standards are deemed too rigid when working in an Agile manner, therefore teams are facing challenges in estimating and planning their iterations.

1.1 Objectives

In this thesis, there are two objectives. The first is to find out if a tool that keeps track of progress will help the team make more accurate iteration forecasts. The second is to determine the factors that influence the progress of an Agile team that are to be considered in the planning phase. The investigation will be done in the context of the Scrum methodology.

1.2 Background

In this section, the theoretical background is presented. The reader who is unfamiliar with the subject is advised to read through in order, to gain a full understanding of the problem statement.

1.2.1 Agile framework

Agile software development is a framework that describes a set of principles, values and patterns that have revolutionized how the software industry thinks and performs in their daily endeavors. Among many of the agile methodologies, Scrum has become the most adopted frameworks across software organizations with many of them who are in the fully agile state while others are in the transition state. The Agile framework advocates flexibility, adaptive change, team collaboration, quick response to change, early and frequent delivery and continuous improvement of product and team efforts. The adoption rate of the agile methodologies has widely increased starting early 2000’s. The Agile term was coined by seventeen software development practitioners who collaboratively published the Manifesto for Agile Software Development [3].

The Agile manifesto laid out the foundation for a successful implementation of Agile practices. It declares the following four values:

**Individuals and interactions** over processes and tools

The individuals involved in a project and the interactions among them are valued more than using processes or tools to drive a development process. Fluid communication and self-organization are examples of how to attain this value.
In the context of process, communication would require scheduling and specific content.

**Working software** over comprehensive documentation

Traditionally, a lot of time is invested in documentation of almost every aspect of software development, for presenting what is needed, and for creating a shared understanding among the development team and the project members. This approach has been heavyweight on the project progress where almost everyone is bogged down with extensive document reading and microdetails. This value does not eliminate documentation in its entirety but rather streamlines it in a form that allows the developer to get started doing the core work of a project effectively.

**Customer collaboration** over contract negotiation

Negotiation is the phase where the customer requirements are laid out. Traditionally, it is done at the beginning of the software development cycle where the requirements are fully collected. Customer is then involved only at the negotiation and the delivery phases. This value favors continuous customer involvement and collaboration during the development phase rather than just at the endpoints of the development cycle. This way, the development team can easier understand the business needs of the customer and ensure that the product delivers what is needed.

**Responding to change** over following a plan

Traditional development methods have regarded change as an expense as a lot of plans are detailed at the beginning. Thus, change was to be avoided once the planning phase was completed. The Agile process takes a different approach. The iterative way of working out the requirements allows for adjustments or even totally new requirements without impacting the project overall. Shifting priorities is a norm and the teams are expected to deliver business value as requested by the customer rather than delivering per plans.

As can be concluded, the primary concerns, the favored values, are critical to success while the secondary concerns are still of importance.

The twelve principles in the Agile Manifesto [4] are used for guiding when applying Agile methodologies to successfully align development efforts with business needs effectively and efficiently. They describe a culture where change is welcome and the customer is in focus. That is achieved by frequent delivery to avoid very long waiting time and by accepting new requests based on the feedback from the recurring demonstrations of the working software. Stakeholder and development team collaboration is also of great importance, likewise direct contact, the face-to-face interactions rather than hierarchical management. It also supports co-location and self-organization of the
development teams to entitle them decision making to have a sense of ownership and to be more productive for higher product quality.

Another thing that characterizes an agile development is the incremental and iterative approach that it takes in the software development cycle. Figure 1 contrasts the distinctions in the traditional and the agile approaches. As depicted in the figure, the traditional approach, here the waterfall method, has a sequential flow through the phases of the life of software development project. During those phases, the plans are followed rigidly without space for adjustments once the project has advanced to the next phase. Emphasis is put on planning, scheduling and implementation which is done all at once with minimal revision through the life cycle of the project. Code and acceptance testing is sometimes planned in a phase on its own, after a deliverable has been produced, not seldom by different persons. Deliverables are in form of documentations submitted and scrutinized by management at the end of each phase.

The agile method on the other side breaks up the software development work into small increments called iterations to minimize the upfront costs of planning and design. An iteration is a short time-boxed period that is typically between one to four weeks and encompasses all the phases that the traditional method runs through. It involves a cross-functional team that undertakes a chunk of work accepted by the product owner that is to be completed during the iteration. While the product might not be ready for market release after a single iteration, the goal is to allow for changes based on feedback from previous iterations and to achieve a working high-quality product with minimal bugs that is releasable to stakeholders.
1.2.2 Scrum methodology

Scrum [5] is an agile methodology that defines a software development strategy for project management and process control. It is a framework that was designed to address complex adaptive problems and it allows the user to employ any set of techniques and processes. Scrum was founded through empirical studies and experience of the authors who are software developers with long time experience in the industry. It provides flexibility to handle the volatility of a software project where the customer is given leeway for requirements change. That is obtained by employing an incremental, iterative approach to optimize predictability and control risk. Scrum has a simplistic approach to deal with project management complexities with intent not to burden the development team with extra management work. This approach allows the team to produce a high-quality product productively and efficiently, that meets customer demands and business needs.

The Scrum framework consists of Scrum teams and their associated roles, artifacts, events and rules where the rules bind together all the elements of the
framework. To successfully implement the Scrum framework, all elements must be well understood and utilized.

There are three pillars that uphold an implementation of Scrum:

**Transparency:** many aspects of the project and the processes must be visible to the team and others involved in the project. Transparency allows for a common understanding of what is being seen and minimizes the risk of own interpretation.

**Inspection:** artifacts and progress must be continuously monitored to make sure unwanted variances are eliminated.

**Adaptation:** the project should be regularly reviewed to avoid divergence from the objectives and be adjusted thereafter to minimize deviations. Adaptation prescribes a set of events that are to be attended regularly by the Scrum team and by the stakeholders if necessary. Those events are explained more below.

### 1.2.3 Roles in a Scrum project

There are several roles involved in a Scrum project:

**Product Owner** is the representative of the customer that is responsible for making sure the project delivers as requested and meets business needs.

**Development Team** is the team that possesses the knowledge to perform the core work of the project. Quite often it is a cross-functional team, meaning that it is composed of persons from different disciplines in software engineering, such as designers, coders and testers.

**Scrum Master** is the leader of the team who facilitates and guides the team and makes sure everybody has something to work on. This role is responsible for obtaining resources necessary for fulfillment of the goals and removing any impediments that keep the team from doing their work. In small organizations, the person undertaking this role may also be part of the Development Team.

**Stakeholders** are the ones who are outside the development team who have an interest in the project. Example of these are funders, project managers and people with direct contact with the customers.

**Scrum Team** consists of the Development Team, the Scrum Master and the Product Owner.

### 1.2.4 Scrum sprint

Scrum prescribes four formal events that are essential for project success. These are Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective. A Sprint is an iteration in the context of Scrum. It is a time-box of typically one to four weeks long but can sometimes be up to two months long depending on the project, although quite rare. The length of the sprint is made short enough to
reduce risk, avoid complexities, enable predictability, control scope and to limit costs to the time-box should something severe happen. The length is also made long enough to be able to produce something useful and to efficiently utilize team capacity.

The goal of a sprint is to provide a useable and potentially releasable product at the end of it. A new sprint starts immediately after the conclusion of a previous one. However, some organizations schedule something called a Lab Day every few sprints and in between, that can last from a few hours to a whole day for employee recovery, to gain some knowledge and to perform other duties not related to the project.

During a Sprint, no substantial changes are made to the plans and the quality goals are not reduced. In the event of huge deviations from the original forecasts, plans are clarified, renegotiated and revised between the product owner and the development team. Mid-sprint work is almost never disrupted; the matter must be brought up between sprints or during the planning or review phases of a sprint.

**Sprint Planning** is the process at the beginning of the sprint that is time-boxed to a few hours where the collaborative effort of the entire Scrum Team is put in to define what the sprint is about. Goals are defined, work is estimated and an execution plan is drafted. The team then self-organizes to undertake the work and coordinate it among themselves. The input to the sprint, the items, is selected from the Product Backlog. A Product Backlog is a list of things that are set out to be completed within a project. The items usually contain short descriptions, setting out the goals and leaving out the details to the team to inspect upon initiation of the work pertaining to the item. Parameters that are considered prior to the process of selection of backlog items, are past performance of the team, projected capacity of the team and the latest increment. The decision as to how much work to undertake is solely up to the team as it is the only group that can assess what it can accomplish.

**Daily Scrum** is a daily time-boxed event which is typically 15-minutes long. It takes place at the same time and place each day to avoid complexity. The whole team should be present but it can start even if parts of the team cannot attend. During this event, work is synchronized among the team by taking turn to discuss what has been accomplished the day before, what is planned to be done in the next 24-hours and if there are any impediments in performing the work.

**Sprint Review** is a time-boxed event, of a few hours, that is held at the end of the sprint where the Scrum Team and the Stakeholders are present to inspect what the team has accomplished. The product is demonstrated, feedback is elicited, backlog is adjusted if needed and attendees collaborate to optimize value produced by the project. The development team discusses what went well, what problems they have faced and how they solved them. Also, work that was part of the sprint that is remaining is brought up. The product owner presents re-
estimated figures and revised plans. Other events such as change in demand or competition, team capability and budget might be discussed. The attendees collaborate to provide valuable information in the subsequent sprint planning.

**Sprint Retrospective** is a time-boxed event of a few hours that is scheduled for the development team alone, after the sprint review but before the planning of the next sprint. During the event, the team inspects its performance and creates a plan for improvement to enact during the next sprint. Elements that are inspected are people, relationships, processes and tools, seeking potential improvements.

The sprint lifecycle is depicted in Figure 2, which starts with the sprint planning, usually time-boxed to a few hours. The result of the planning phase is a Sprint Backlog, that is a set of things that are projected to be completed by the end of the sprint. The next phase is the execution of the plan where the bulk of the work of the sprint is carried out. The sprint ends at a set out date, whether all items were finished or not. Afterwards, the scrum team demonstrates to the stakeholders the results of the increment, the sprint along with the previous results if necessary. The team then gathers together, reviews its performance and decides how to improve it in the next sprint.

1.2.5 Scrum artifacts

In a Scrum project, artifacts are produced that are visible to all project participants to make sure a common understand is shared and that inspection
and adaptation are well versed into the project. They are the Product Backlog, Sprint Backlog and the Increment.

**Product Backlog** is an ordered list of items that defines what is needed in a product. It is a single source of all features, functions, requirements, enhancements and fixes that constitute the changes to be applied on the product in future sprints. It is the responsibility of the Product Owner to construct and maintain the product backlog in accordance with the business values of the customer. The product backlog is not constructed in its entirety from the very beginning. Rather, the construction is an evolutionary process where the customer provides feedback and requests changes and the product is adapted to the environment where it is to be used. A backlog item has the following attributes: description, order, estimate and value. An estimate is the amount of effort required to complete an item. Items that are ordered higher are often smaller in size, more detailed and have more accurate estimates as they are to be enacted sooner than lower ordered ones which are often large in size and roughly estimated, as depicted in Figure 3. The value of an item is the business value that would benefit the customer. The product owner makes sure to prioritize items with highest value first. Value and size often do not correlate, meaning a large item does not necessary mean larger business value than smaller items.

*Figure 3. Product backlog of a Scrum project visualized. (Source: http://www.informit.com/articles/article.aspx?p=1928232&seqNum=3)*
**Sprint Backlog** is the set of items that are selected from the Product Backlog to be completed during the sprint. It is a forecast by the development team on how much work it can deliver at the end of the sprint. The Sprint Backlog also includes a plan on how to carry out the completion of the items to realize the goal of the sprint and deliver the new increment to the customer. From the perspective of the Product Owner and the stakeholders, it is a real-time picture of what is ongoing in the project. The Sprint Backlog is not static; as with the Product Backlog, it emerges throughout the sprint as the team works through the plan and learns more about the items.

The selected items in a sprint are then decomposed into maintainable tasks that correspond to the technical terms. This is done by the development team itself, in contrast with the items which are created in a collaborative effort between the customer, product owner and the development team. The set of tasks can then be carried out by any member of the development team without much ambiguousness.

![Diagram of Backlog hierarchy in a Scrum Project](https://www.scrumalliance.org/community/articles/2014/november/csm-workshop-key-takeaways)

Figure 4 depicts a hierarchical view of the different backlogs and lists of items, in a scrum project. The product backlog is the largest, which contains everything that is projected to be completed during the project. A scrum project is usually made of two or more sprints. At the beginning of a sprint, a sprint backlog is
created. A sprint backlog is a subset of the product backlog. A list of tasks is then composed for each sprint backlog item, to be executed to complete an item.

An **Increment** is the sum of all completed items from the Sprint Backlog plus the value of the increments of previous sprints. The increment should be potentially releasable although the product owner may choose not to.

---

**Figure 5. Scrum process in a software development project. (Source: https://commons.wikimedia.org/wiki/File:Scrum_process.svg)**

The Scrum process is summarized in Figure 5. A scrum project is initiated by construction of the product backlog in collaboration with the customer. Throughout the project, the product backlog is refined according to the results of the sprints and the feedback and the requests from the customer. In each sprint, a sprint backlog is constructed and worked upon during the sprint. A daily scrum, also referred as a Standup Meeting, of usually no more than half an hour, is held at the start of the day to discuss what has been completed in the previous day, what is to be done during the day and eventual obstacles. At the end of a sprint, a new increment is created. The scrum process iterates with a new sprint until the product backlog is exhausted or certain criteria set out between the management and the customer is met.

### 1.2.6 User stories

The backlog items mentioned in the previous section can take on any form. The Scrum framework does not specify any format or criterions to use in a Scrum project. But what is most common though is to form the items as User Stories [6] that entail a high-level description of the requirement from the customer perspective, hence the name. This way, transparency as a Scrum pillar is put into the focal point of the project, since the user stories are written in plain English that is understandable by the scrum team and the stakeholders. It starts out as a
conversation between the participants around the story and the mutual understanding of what to build, the value for and how the end-user will utilize the value will concretize the final description. Acceptance Criteria and Definition of Done are sometimes also discussed. Acceptance Criteria is specific to a user story that lays out the requirements to be met for the user story to be accepted by the customer. Definition of Done is also a list of requirements but that applies to any user story and is usually agreed upon in between the scrum team. There is no partial acceptance to either of the requirements, either everything is fulfilled or the results are not accepted. A user story only describes a feature, it leaves out any technical aspects and how to carry out the implementation.

A User Story can take on the following form:

As a <type of user>, I want <goal, objective>, so that <benefit, value>

The formulation does not have to be in that particular way, but it helps the creator articulate the following questions:

Who are we building it for? As a <type of user>

What are we building? I want <goal, objective>

Why are we building it? So that <benefit, value>

User Stories are often written on post-it notes that are then stuck on a whiteboard that is accessible by everyone involved in the project. The whiteboard has the advantage that one can draw lines between the notes and write clarifications that would otherwise not be as doable on a piece of paper, as mistakes cannot be easily reversed. Dependencies among user stories are made as few as possible and the stories are laid out in a flat list. That is to keep the work coordination overhead as low as possible and to be able to model the work on the board with ease.

Once an initial set of User Stories has been constructed, each story is then assigned a value that corresponds to the business value of the customer. That is done in collaboration between the customer and the product owner. The product owner is then responsible to prioritize and make sure the development team works on the relevant items that provide most value for the customer.

1.2.7 Estimation techniques

A backlog item has several attributes, one being the estimate of size or effort. Teams are usually interested in the duration of the items and the project, for planning and scheduling purposes [7, pp. 33-34]. Once the effort of an item has
been estimated, the estimate of duration can be derived. The team’s rate of progress, called velocity, is used to calibrate effort estimations to time.

There are several factors that may influence the estimation of effort [8]:

**Amount of work:** the size of the work to be accomplished. More work does not automatically translate to a higher estimate on a one-to-one ratio. One must consider economies of scale, as less additional effort may be required to complete other parts of the work.

**Complexity of work:** how much thought must be put into the work. The higher the complexity, the greater chance there is of a developer mistake and consequently more effort needed.

**Risk in doing the work:** the feasibility of doing the work. One might be forced to redo other parts of the system, unanticipatedly.

**Uncertainty in doing the work:** the customer might be unclear on what is being requested, perhaps due to insufficient information in supporting of what is being requested.

There are two common methods of estimation, Story Points [7, pp. 35-40] and Ideal Days [7, pp. 43-47]. Only the former will be described here, as being used throughout this report. The reason is that there are more benefits of using the former, in the long run, as discussed by Cohn [7, pp. 69-75]. Story Points is a measure using simple numbers that is solely used to denote effort. Estimation starts out by selecting a rather simple user story that will act as a baseline which is assigned a number, usually in the low range, and then used as a reference point for estimating the other user stories. Any series of numbers can be used for scaling the user stories but most common is to use the Fibonacci sequence (1, 2, 3, 5, 8, 13, ..) or the sequence in Power of 2 (1, 2, 4, 8, 16, ..) [7, p. 52]. The reason for an exponentially growing series rather than a linear one is that small user stories can be more accurately estimated, hence the scaling at small numbers is more fine-grained. While larger stories exhibit higher uncertainty and risk and so cannot be estimated as easily; there is no use of fine-graining the scale at large numbers. The scaling can be seen as bucketing where items of roughly the same size are put in the same bucket. The law of diminishing returns applies also in estimations, putting too much effort into it will yield less results and there is also the risk of the customer changing his mind. [7, pp. 49-51]

Estimation by Story Points does not define the duration of the work. All what is described is the effort that is needed to complete the work, relatively to the baseline. An item that is twice as much of another item should be given double the value of the referenced item. The duration of the work depends on the rate of progress of the developer. Two developers might agree on the size of the item but may not necessarily agree on the duration of the work. Thus, Story Point is a relative measure, the raw value is unimportant. Often, user stories are compared
to more than one story, one smaller and one larger, to produce finer results, this method is similar to triangulation.

The three most common techniques for estimation are [7, pp. 54-56]:

**Expert opinion**: an expert is asked to assess the size of a user story. This is a quick way to get an estimate, but is less useful in an agile project. A user story is formulated from the perspective of the user and the functionality usually stretches cross many disciplines which the expert may not possess.

**Estimating by analogy**: user stories are estimated by comparing them to an assortment of already estimated stories.

**Disaggregation**: large user stories are split to be no more than a few orders of magnitude in size than already estimated stories. That way estimations can be more accurate if most stories are on a small range.

Planning Poker is a common consensus-based estimation method where the collaborative effort of the Scrum Team is utilized. The session starts out by the product owner explaining the user story being estimated. Team members then discuss and ask questions. Afterwards, each member picks a card representing the estimation he thinks fit the story, without disclosing the choice. At the end, everyone reveals their card and the lowest and highest card holders got to explain their choice. It might take a few rounds until there is a consensus on the estimation, by reselecting their cards. The session may be deferred to another time if additional information is required. Planning Poker subsumes all the techniques mentioned above. Scrum teams are cross-functional and given that, all members possess something unique that contributes to the assessment of the effort needed. Estimation is done by analogy, comparing to other stories and large stories are often split to better fit the size of the bulk of stories.

**1.2.8 Sprint forecasting**

Sprint Forecasting is defined as the estimation of the amount of work that is to be scheduled and committed to in the upcoming sprint. It is a separate process from the estimation of the stories where the size of the story is determined. In forecasting, the size of work that the team can deliver in the upcoming sprint is determined. The body of work that is to be committed to is measured in the same unit as user stories. It is the sum of all stories that have been adopted to the sprint backlog. Usually a total of story points is computed and stories are then selected until the quota is filled. There is no predefined procedure for estimating and forecasting, however it is recommended that first a handful of user stories, that are to be acted upon in the next few sprints are estimated. The reason is to avoid any bias or story point inflation, where team members are tempted to assign a larger size to fill up the quota faster, hence reducing the work load.
1.3 Aim and purpose

The purpose of this thesis is to investigate the applicability of a tool that keeps track of scrum team progress and whether it will improve predictability in sprint planning. Factors that influence the forecasts in a sprint will be studied to propose enhancements to the tool. To formulate it as a question: Does a tool assist a scrum team and improve predictability in sprint planning? What factors influence the decision on how many story points to enact? The aim is to be able to create accurate and reliable plans to bring predictability in a project, so that related activities can be scheduled accordingly and plans such as budgeting become more accurate.

In the implementation of the tool, the following definitions will be used:

**Sprint Forecast** is the estimated number of story points that are to be committed to in the upcoming sprint.

**Sprint Actual** is the outcome that is computed at the end of the sprint when the development work has been concluded. It is the sum of story points of all completed user stories.

**Aggregated Velocity** is an average value of historical sprint outcomes that pertains to a sprint. It can be computed using any model such as SMA (Simple Moving Average) or EWMA (Exponentially Weighted Moving Average). Here, SMA is used.

**Velocity Factor** is the Sprint Actual in relation to a time unit, usually days. The time is the team capacity entered in the planning phase of the sprint.

The following formulas are used in the computations in the tool:

\[ v_n = \frac{a_n + a_{n-1} + a_{n-2}}{3} \]

Where \( v \) denotes Aggregated Velocity, \( a \) the Sprint Actual and \( n \) the sprint id. This is SMA(3), the average of the Sprint Actual of the current and the past two sprints. Aggregated Velocity will not be used elsewhere besides allowing the team to track it and drawing own conclusions.

\[ v_{f^a_n} = \frac{a_n}{t_n} \]

Where \( v_f \) is the Velocity Factor, \( a \) is the Sprint Actual and \( t \) is the time (number of days in a sprint). The subscript \( n \) is the sprint id and the superscript \( a \) is the actual, denoting that it is computed based on historical data.
\[ v_{fn}^e = \frac{vf_{n-1}^a + vf_{n-2}^a + vf_{n-3}^a}{3} \]

Where the Velocity Factor is being estimated for the upcoming sprint, denoted by the superscript \( e \) for estimated. It is computed based on the actual Velocity Factor from past sprints.

For all computations, SMA(3) has been used for consistency.

The suggested forecast is then computed as the following

\[ forecast = v_{fn}^e \times t_n \]

Where \( t \) is the number of days in the sprint.

Forecast Accuracy is computed as a percentage and as a difference.

\[ \frac{\text{Sprint Actual}}{\text{Sprint Forecast}} \quad \text{and} \quad \text{Sprint Actual} - \text{Sprint Forecast} \]
2 Method

The approach used here is threefold. First, a survey was created and sent out as a pre-study. Then, a web-based tool was developed in accordance with the guidelines given by the Agile coaches at Agiltec Group AB. Lastly, a literature study was conducted in depth to find out how to address common issues in agile team progress and enhance the tool.

2.1 Project outline

Initially, coaching by certified Agile consultants on basic terms and metrics was completed along with discussions around the technical aspects of the tool that had been envisioned. The terms and metrics that the project ultimately settled down to are explained in section 1.3. It was set out to use Scrum as the development methodology for the tool. Development planning commenced by construction of the product backlog using USM (User Story Mapping) [9], which is explained in section 2.5. The sprint length was set to two weeks and three recurring sprint events were planned, at the same place and time. Planning, refinement and review. Sprint Refinement occurred mid-sprint to measure progress against the plan. Sprint planning and sprint review have been explained in section 1.2.4.

The development of the tool, which is explained in section 2.3, was started almost immediately in the beginning of the project. The knowledge of the coaches at Agiltec Group AB was used due to time constraints of the thesis. The tool was to be released to test groups who then had sprints that consume a vast amount of time relative to the time at hand. The aim was that on average, each team would have run 3 or 4 sprints before evaluation. The evaluation is explained further in section 2.4.

2.2 Survey

A quick survey was created and sent out to agile teams in the local software industry. It is a pre-study for the research with the purpose of gaining insights into the problems the teams are facing and having a more detailed view of what challenges they are experiencing. The survey is listed in appendix 6.1.

2.3 Tool development

The tool is web-based and was developed through collaborative work with the stakeholders. It consists of two major parts, the front-end and the back-end. Two development environments had been setup, a staging and a production environment. The former was used to deploy bug fixes and new features for testing purposes and the latter was used by the end-user. The front-end was developed by a fellow student doing his master’s thesis in Interaction and Design. It consumes data from the back-end and formats it appropriately for the user.
2.4 Evaluation

To evaluate the tool, it was released to Scrum teams at the IT division at the Swedish Tax Agency (Skatteverket) in Umeå, Sweden after four sprints, which would be eight weeks. Bug fixes and additional implementations to support the needs of the teams were applied. Some teams had different sprint lengths and unsynchronized sprint timings but on average, a team had a three-week sprint. The teams then had some time to utilize the tool in their planning process while the literature study and the report writing was taking place. Members from different teams were then interviewed to identify the usability of a such tool in Scrum planning and assess whether planning accuracy had improved. The interviews were more semi-structured as explanations were given and more clarifying questions were posed based on initial responses. Different teams have different experiences; the initial set of questions might not be easy to answer from first glance. The interview was carried out by having a laptop in front of me with the questions, asking the interviewees the questions in turn and noting down the responses verbatim. The interviewees were not using any devices at the moment, mostly focusing on the questions.

2.5 Research

An in-depth study of existing literature was conducted to find out the major factors that influence plans and progress in a Scrum team. The purpose was to complement the first part of the work where a basic implementation was done. Further information was sought to be able to make more intelligible decisions for Scrum teams in the planning phase, when using the tool.

2.6 User story mapping

USM is a method used for requirements documentation, that surrounds the discussions from the perspective of the user. It is not an alternative to writing down user stories but it is an elaboration on the intentions of the user, to attempt to capture the ultimate goals of the project. In traditional flat mapping of user stories, origins of each story are lost as discussions progress further. There is no way to track the origins of the user story to identify the context and the business needs of the user. USM enhances the backlog by layering it into a hierarchy where the leaves are the User Stories the agile community is familiar with while items at a higher order can be used for descriptive purposes only.

In Figure 6, the pink notes are the user stories and the orange notes are the activities meaning the ultimate goal of the stories beneath each activity. Here, the mapping or the hierarchy consists of the activities and the user stories each activity subsumes. The yellow notes identify the sprints and the blue notes specify the goal of each sprint. The light blue notes apply technical attributes to the user stories, that would otherwise be part of the user story description. The attributes were exported to own notes for structural convenience only. In this
case, the attributes define the access permissions needed for each story below the permission note. User stories at the top are usually essential capabilities and are the backbone in the system. Stories that are lower in the hierarchy are the skeleton and they are prioritized among each other under each activity. Stories are then selected from each activity, horizontally in Figure 6, to form a MVP (Minimum Viable Product). That constructs a running skeleton as the project progresses and new sprints are created.

*Figure 6. User story mapping for the development of the tool.*
3 Results

In this chapter, the survey results, the tool architecture, the tool evaluation and the in-depth study are presented. The survey template is found in appendix 6.1. The sub-sections can be read in an arbitrary order.

3.1 Survey

14 responses were received. The survey was made anonymous so tracking to identify the respondents is not possible. Only the integral part of the results of the survey will be presented here.

Nearly 79% of the respondents consider sprint planning to be a challenging task. All respondents consider estimation of user stories as a challenging task. 64% of them have found assessing team capacity, the number of story points the team can cope with, as a challenge. 21% of the respondents find that team related issues such as communication and work distribution hinder team progress. 21% find that managerial problems, the communication between the master roles (Product Owner, Scrum Master, Linear Organization) to be the real culprit in team progress. 29% often experience their projects to be running late. 50% experience that they often spend time fixing bugs and 35% spend their time doing administrative work that does not provide direct value to the project. 29% find that there is often too much in progress in a sprint while 50% are often interrupted by the stakeholders. Only 7% find that there are large discrepancies between forecast and progress in a sprint.

The results are summarized in the table below.

<table>
<thead>
<tr>
<th>Challenge/Problem</th>
<th>Respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation of user stories</td>
<td>100%</td>
</tr>
<tr>
<td>Sprint Planning</td>
<td>79%</td>
</tr>
<tr>
<td>Forecasting team capacity</td>
<td>64%</td>
</tr>
<tr>
<td>Often spending time fixing bugs</td>
<td>50%</td>
</tr>
<tr>
<td>Interruptions by stakeholders</td>
<td>50%</td>
</tr>
<tr>
<td>Spending time on things with no direct value</td>
<td>35%</td>
</tr>
<tr>
<td>Project often running late</td>
<td>29%</td>
</tr>
<tr>
<td>Too much in progress in a sprint</td>
<td>29%</td>
</tr>
<tr>
<td>Communication and work distribution</td>
<td>21%</td>
</tr>
<tr>
<td>Managerial problems</td>
<td>21%</td>
</tr>
<tr>
<td>Large discrepancies between forecast and progress in a sprint</td>
<td>7%</td>
</tr>
</tbody>
</table>

Regarding the tool which purpose is to aid an Agile team in the planning phase, the following ratings were observed. The scale ranges from 1 to 5, where a larger number indicates larger value. In the diagrams below, the rating percentages for each feature are displayed.
3.2 Tool architecture

A back-end service that exposes a RESTful API [10] was developed using Spring Boot framework [11]. The framework provides IoC (Inversion of Control) [12] for dependency handling in the application and support to create a RESTful service to expose endpoints for network applications. The framework also facilitates database access management. The chosen communication data format is JSON for its readability and compactness. For storage, the MySQL relational database management system [13] is used.

The application was designed to be stateless, meaning no session data is preserved on the application server and the database is the sole mean of data storage, to keep the development complexity as simple as possible and to promote its scalability. Scalability makes it easier to deploy an application in a cloud environment [14], which can handle demand fluctuations to the service either elastically or manually. Increasing demand could originate from the usage by new teams and organizations. The load can be balanced by a load balancer, following the front controller pattern. It should be noted though that the storage service (the database) must also scale as well to not bottleneck the system. Several cloud providers accommodate the need of scalable storage and load balancing without any additional setup.

The application can be built into a deployable unit, a JAR (Java ARchive) file, using any build tool and deployed anywhere the Java Runtime Environment 8 is supported. In this work, Gradle was used. The JAR file is regarded as a fat JAR as all libraries and engines needed are embedded into the JAR, no additional setup is required. The JAR file can be easily setup for handling by the operating system as a service that is run in the background.

The front-end of the tool utilizes the framework Angular 2 [15], which revolutionizes how websites are built. Applications are downloaded to the client web browser and pages are rendered locally by just retrieving the required data from upstream. Applications are often SPA (Single Page Application) meaning there is no extra page loading in the browser. The DOM (Document Object Model) is manipulated to display different components.

Due to the generality of the back-end service, where RESTful endpoints are available, it could be complemented with any client which has web access. The back-end does not enforce or constrain the user to any specific platform. Though it cannot be used on its own, as it does not provide a visual interface. This type of design supports ubiquitous computing where the user can access the service from any endpoint, be it on a desktop or mobile and it does not matter where the user is located.

What follows next are snippets of the view of the tool to demonstrate how it looks like to the user and the capabilities provided in the application. Only the essential parts of the tool, which are relevant to the study are shown. The other parts such
as user, team and project management have been left out, although they are essential in production use.

![Sprints](image)

**Figure 7. List of completed sprints.**

In Figure 7, a list of sprints is displayed in a panel along with filters and action buttons. The user can select a sprint and enter data as shown in the other figures. Each sprint has a status and action buttons to update and delete a sprint. A status color legend is shown above the panel.
Figure 8. Available time input for all team members for a sprint.

In Figure 8, the time input window is displayed. This window allows the user to input available time in hours, the team capacity, along with the time-boxed time for other activities. The current implementation regards the available time only. Future extensions might make use of time-boxed data too.

Figure 9. Forecast input for a sprint.

In Figure 9, the user is presented with a suggested number of story points for the specified sprint. The user is also prompted to enter the team’s forecast for the sprint.
In Figure 10, the user can enter the sprint result which is the Sprint Actual (marked as Actual in the figure) and a comment for providing more information about the sprint. The comment is for consumption by the team only. The tool does not translate it into something useful for internal use. The user can also change the status of the sprint to finished, to make it a part of the historical data. Metrics such as Aggregated Velocity (marked as Velocity in the figure) and Forecast Accuracy are also displayed.

Figure 10. Sprint results input and metrics view.

Figure 11. Forecasts plotted against actuals for all completed sprints.
In Figure 11, plot graphs of the forecasts and actuals of completed sprints are drawn to visualize the discrepancies among them (sprints are shown from oldest to the left, to the newest to the right).

**Figure 12. Velocity plotted for each completed sprint.**

In Figure 12, the team’s velocity is visualized over time. A team observes its velocity, its progress and challenges itself to continually improve it.

**Figure 13. Accuracy which is the difference between forecast and actual is plotted for each completed sprint.**
In Figure 13, an accuracy curve is displayed. A positive value indicates the team has done beyond than expected. A negative value means the team completed less than anticipated. The aim is to have an accuracy close to zero, since then plans are stable and more reliable.

3.3 Tool evaluation

Interviews were conducted with the users of the tool, the IT-development division of the Swedish Tax Agency in Umeå, on 7th June 2017. Representatives from three teams were interviewed in turn where each representative acted as a spokesman for the whole team, voicing their feedback and responses. Each interview lasted around 30 minutes. It would have been much costlier to hold an interview with each member of the teams. The questions and the answers can be found in appendix 6.2. Each bullet point under each question is a response from one of the representatives. The order of the responses from the respondents is the same for all questions.

3.4 In-depth study

Factors that may influence the decisions in the planning phase and impact the progress and the sprint outcome can be split into two major categories. Historical and present data that are mainly of technical and social character. The history is a trail of actions (and results) that have been carried out in the project by the team and the organization. It is a hint on what is to follow and the direction of the progress path. The technical aspect encompasses team, organization and project parameters. These parameters are subject to constant change due to internal and external events and are to be reevaluated during each planning session to account for the changes. The social factors are less explicit and are less likely to change but must nonetheless be considered to accommodate the projections in a project to its environment.

Below, certain aspects of a scrum project and its environment will be outlined. These aspects encompass actions to be taken at various stages of a project, which are proven to yield better results and consequently more reliable estimates. Aspects that will be detailed are team performance, social factors, organizational patterns, user story estimations, estimation procedure improvement and sprint length.

According to Downey and Sutherland [16], fluctuating performance needs to stabilize to increase predictability. There is a need for a set of metrics and constraints to be able to systematically achieve high productivity. Scrum must be seen as an ecosystem of interdependent parts that are to be monitored using a set of metrics and coordinated by the team through daily inspection. Lately, agile teams are having trouble measuring performance. Global surveys conducted by Downey and Sutherland show that 50% of the teams do not know their velocity
due to difficulty of measuring and improving the rate. The proposed measures by the authors to improve the concerned issues are the following:

- Shift the focus in the daily standups from the individual to the sprint backlog to help the members function as a team. As the sprint backlog is put into focus, discussions of items from other specialties are prompted, hence aiding in cross-training the team. This in turn will quicken the team’s learning allowing it to move through the phases Forming, Norming, Storming and Performing rapidly.

- Team should spend minimum time on digesting work and maximize the time spent on achieving the work. User stories that are to be enacted should fit the INVEST criteria [17], which stands for:
  - Independent – work can be started immediately, there are no dependencies.
  - Negotiable – it is not totally explicit, it can be brought up for discussion.
  - Valuable – it must bring value to the stakeholders.
  - Estimable – the effort to complete it must be quantifiable.
  - Sized to Fit (Max of ~50% of Velocity) – be small enough to allow work dispersion and allow planning with accuracy.
  - Testable – it must contain necessary information to enable test development.

Downey and Sutherland propose the following metrics to monitor team performance:

**Velocity** – sum of original estimates of applied work (finished user stories)

**Work Capacity** – sum of all work applied toward user stories, whether finished or not

**Focus Factor** – Velocity/Work Capacity

*Focus factor should be around 80%. Too high and the team capacity is being underestimated in order to appear perfect or are ignoring other organizational duties. Too low and the team is often disrupted by external events or otherwise unable to turn their work into accepted work.* [16]

**Percentage of Adopted Work** – extra user stories that were adopted from the product backlog into the sprint backlog because the team had early finished the original forecast. It is computed by dividing the original estimates of adopted work with the forecast for the sprint.

**Percentage of Found Work** – the extra work that must be completed in order to deliver the user stories. It is computed by dividing the estimation of extra work with the forecast for sprint.
Accuracy of Estimation – \( 1 - \frac{\text{Sum of estimate deltas}}{\text{Total Forecast}} \)

An estimate delta is the difference between the original estimate and the re-estimated size of effort. Accuracy of estimation should be around 80%. Too high and the team is over-planning by doing too much research and digestion. The team should accept more risk and spend time being productive rather than sitting in meetings studying what they are trying to do. Too low means the estimates are very inaccurate. This could be the result of miscommunication within the scrum team, that there is a misunderstanding of what is being requested, the team does not possess the competence for the technology they are being asked to build, that the requirements are changing midst of a sprint or that there is a dysfunctional group dynamic where members are not sharing knowledge among the team. [16]

Accuracy of Forecast – \( \frac{\text{Sum of Original estimates}}{\text{(Sum of original estimates + Sum of adopted work + Sum of Found work)}} \)

Forecast accuracy that is too high could mean the team is under pressure from above and the team is trying to achieve perfect metrics. Perhaps too much time is spent on work digestion. Too low and the team may not be adequately protected by the scrum master in an environment where multiple teams are planning at the same time. The coordination between the product owners may be malfunctioning and the team is being loaded an unplanned quantity of work. [16]

Targeted Value Increase (TVI) – \( \frac{\text{Current Sprint Velocity}}{\text{Original Velocity}} \)

Success at Scale – Each point on the Fibonacci scale has a ratio of accepted attempts of user stories to the total number of attempts. The team should pick stories that have sizes they have been successful at. The team should not be denied the opportunity to attempt large stories but they should be well advised by the scrum master before proceeding.

Win/Loss Record – Each sprint is a win if 80% of the original forecast is accepted and the sum of found and adopted work is less than 20% of original forecast.

Scrum is usually explained as a set of principles and methods that are easy to implement. This has led to the belief that most projects should achieve success with the mere setup of Scrum. Analysis conducted by Coplien and Harrison [18] shows that Scrum is more complex than previously perceived. The analysis reveals that Scrum incorporates a set of organizational patterns, that require a considerable effort of the organization towards Scrum to achieve success. Scrum itself did not initially account for every problem out there, but as it evolved and improved over time, several practices have been adopted to address common problems.
A group of scrum researchers identified the following organizational patterns/routines [19]:

**Stable Teams** is a more of a requirement than a pattern. It states that to achieve high performance in a team, the members of the team must remain the same from one project to another. The team should also be small where the optimum size is five members. In a small team, the communication path is simple which allows for communication saturation.

**Yesterday’s Weather** suggests a number of story points for the upcoming sprint based on the result of the last sprint or an aggregate of historical data. The more recent the data, the more reliable predictor it is.

**Swarming** is a way of processing the user stories. It states that teams should swarm around a single user story and give it their maximum effort to get it done as soon as possible. This solves the problem of many teams who struggle to finish a sprint because there is too much in progress. The person that picks up the item is the captain and everyone must help the captain if they can and he must not be interrupted.

**Interrupt Pattern** states that a team should allot time for interrupts by the stakeholders. Interrupts are requests to alter the adopted user stories or to re-prioritize stories being either on the sprint or the product backlog. The allotted time is usually defined as story points, as a fixed slot or as a percentage of the team’s velocity. Requests must go through the product owner for triage. Research shows that a team does significantly better if it plans for interrupts than a team that does not, even when no interrupts are experienced. [20]

**Daily Clean Code** states that a team should strive to have a clean code base at the end of every day. Bugs and issues that arise should be fixed before the end of the day. Research shows that a bug not fixed the same day it is created could take as much as 24 times longer time three weeks later.

**Scrumming the Scrum** is the identification of the largest impediment in a sprint and then composing it as a story to be processed in the next sprint. The story is to be enacted and evaluated just like any other story, reviewed during sprint review. Impediments can endanger the success of sprints and drag down a team’s velocity.

**Happiness Metric** is a predictive indicator of how well the team is doing. If the team is not feeling well about the organization or the roles, or if there is a major roadblock then that can impact the team’s velocity.

There is also the social aspect when forecasting the capacity of the team for a sprint. Lenarduzzi [21] conducted a study to analyze the social factors that influence the software effort estimation. Factors that affect the effort required to complete a story also impact the team’s ability to complete a set of stories. There
is a correlation between these two tasks, the software effort estimation and sprint forecasting. Therefore, the results shown are applicable to some degree in the planning phase of a sprint.

Vidgan and Wang [22] consider the software development activity as a complex adaptive system. The activity is adapted to its environment which has the interaction of the team members as the low-level component. Therefore, the social factors are of high importance to be considered in an agile project, as the activity is human-centered. It is stated explicitly in the agile manifesto [3] that individuals and interactions are to be preferred over processes and tools. Also, “customer collaboration over contract negotiation” is one of the agile principles.

Lenarduzzi [21] concluded that the following factors are very important:

- Work pressure
- Communication
- Work dispersion

The following factors are moderately important:

- Language and cultural differences
- Communication process
- Experience from previous projects
- Working time
- Domain knowledge
- Team structure
- Technical ability

While the following factors are of little importance:

- Competence level
- Familiarity in team
- Managerial skill
- Familiarity with the project

Chagas et al. [23] conducted a study to find out the impact of human factors in agile projects and which ones are highlighted by the software industry. The results show that for a successful team and project, the following factors are vital to consider:

- Communication
- Collaboration
- Autonomy
- Client involvement
- Trust
- Motivation

The human factors that are often highlighted by the industry during the development of agile projects are the following (in descending order of importance):

- Communication
- Collaboration
- Trust
- Motivation
- Experience
- Leadership

Pirzadeh [24] defines the social factors into three categories:

Individual – related to the characteristics of the individual such as personality.

Interpersonal – related to the factors among the team members such as communication, cooperation and team work.

Organizational – related to the upper management and decision taking by the stakeholders.

Pirzadeh concludes that the individual factors are the ones that present most influence in an agile project.

Another source for sprint plan deviations is the uncertainty that emanates from the dynamic nature of an agile project, where user stories are added, revised or removed through the life of a project. According to Popli and Chauahn [25], this leads to poor estimation of effort for the completion of the user stories. Uncertainty cannot be eliminated completely but it could be reduced to manageable levels. To attain that, user stories are broken down into tasks or sub-stories at a granularity with a tolerable level of uncertainty. The fewer elements that must be considered in a story, the more reliable the estimations. Each sub-story is then estimated using three measures.

1. The minimum number of story points required to complete the story, assuming all resources are available and predecessor stories are completed.
2. A quick estimate, which is often the average case.
3. The maximum number of story points needed to complete the story.

An estimate for each sub-story is then calculated by weighing the quick estimate as a four and the other estimates as a one then averaging them all. The estimate of the original user story is then the sum of all sub-estimates.
Uncertainty in the estimates of the user stories can also be handled by evaluating the estimations themselves. Raith et al. [26] propose a few metrics to consider when running a planning poker session. The variance in an estimation round, the number of estimation rounds and the duration of an estimation. The variance is computed by substituting the estimation scale by position numbers e.g. (1,2,3,5,8,13,20) to (1,2,3,4,5,6,7) and using the estimates of all participants and the final estimate as expected value. The number of estimation rounds is also a good indicator, most estimation sessions end in two rounds. A longer estimation session is an indicator of unreliable estimates. The duration of an estimation session is compared to the average duration of previously run sessions, a higher value indicates less reliable estimates.

The sprint length is also a parameter that affects the success of a project [27]. An excessively long sprint will reduce the opportunities for feedback from the customers and make it difficult to deal with specification changes. Sprints that are too short will increase the overhead cost of planning and reviewing. That may push the developers to break down the stories to make them more manageable thus increasing integration costs. Two important parameters in a project to consider when deciding the sprint length are the variety and the complexity. Variety is the probability for a change in the requirements. Complexity is the probability that dependencies among the requirements are generated. Dependencies can generate integration costs and the cumulative effect can get very huge. It has been established that a high variety requires short iterations due to the increased demand from the customer and more planning is required to reduce integration costs. A high complexity requires longer iterations to deal with the dependencies, hence reducing integration costs.
4 Discussion

In this chapter, the survey is reviewed to discuss how the issues agile teams encounter can be resolved and proposals on how to enhance the tool are presented. Threats to the validity of the research conducted and future work are also discussed. The sub-sections can be read in an arbitrary order.

4.1 Survey review

Many of the users, almost 3 out 4, found it challenging to plan their sprints and furthermore make accurate forecasts. Most of the tool evaluation respondents claim that the tool helped them out in these critical issues. Some users were already using similar measures as those incorporated into the tool, so they had experience in such matter. The conclusion is that it does improve the accuracy.

Most of the users claimed they had problems accurately estimating the effort of the user stories. As the tool is disconnected from the backlogs, it is not possible to assist the user in this matter. However, in section 3.4 there are certain steps that are proven to produce substantially improved results, given the prerequisites are fulfilled.

Some of the users claimed they had been spending too much time fixing bugs. It is not stated whether the problem bottoms in bad coding habits or whether the developers are taking time to fix the bugs. If the latter, the users are encouraged to implement the Daily Clean Code pattern, as described in section 3.4. It has been shown that the longer the bugs persist, the harder it gets to iron them out, which translates into longer debugging sessions. [19, p. 4725]

Many of the teams were experiencing frequent interrupts by the stakeholders. It is advisable that the teams implement the Interrupt Pattern [19, p. 4724] as described in section 3.4. Also, the scrum master and the product owner should take an active role in shielding the team from disturbances and unimportant requests.

Many users stated that they often have too much work in progress. They feel they are burdened as many stories are started on and are to be handled concurrently. One cause may be that the team size is too large since simple communication is not easy and the team is winding up with too many in-progress stories. To mitigate this issue, an optimal team size must be found, following the advice found in section 3.4. The users could also be over-optimistic and are overrating their own capabilities or that they are under pressure from the management. The Swarming Pattern [19, p. 4724] could be a solution for these problems. The users are encouraged to cooperate more and start on new stories when other team members have been consulted first. Another issue is the general unawareness of the INVEST criteria [17], explained in section 3.4, which is clearly depicted in the responses of the survey. Sprint backlog items that do not fulfill the INVEST
criteria may cause issues and endanger the successfulness of the sprint. Also, users may be attempting stories of odd sizes; stories that are too large that the team have not been successful at such sizes before. The Success at Scale pattern could help when determining whether stories should be enacted or revised and re-estimated.

4.2 Tool enhancements

In this section, various suggestions for improvement of the tool will be brought up, based on the in-depth study that was conducted and the feedback from the users of the tool. Though what is to follow is mostly based on research, further evaluations may be required to establish credibility in the tool. The technical aspect will be acknowledged for the most part.

One factor that should be tracked and perhaps accounted for, is the Work Capacity, which is all the work applied during the sprint, regardless if it has led to finished stories. By considering this factor in the unit of story points, one could compute the Focus Factor, which is the percentage of the effort applied towards the completion of what has been adopted into the sprint backlog. The current factor for sprint outcome is the sprint velocity, the finished work, which should be renamed to avoid confusion. The reason for the adoption of that factor, is for a team to be able to track its effort and healthiness.

Factors that are advised by agile authors to track, are the adopted and found work. Adopted work is the extra body of work that has been adopted into the sprint backlog. It is already incorporated into the tool, under the factor named Accuracy. Accuracy accounts for both extra and remaining work, while adopted work accounts for extra work only. Found work is the unanticipated extra work that must be completed to accomplish the goal of the story. No extra steps are deemed necessary for the improvement of the tool in this matter, as the issue of misjudgment of the body of work is mitigated at the estimation stage. Necessary steps for the alleviation of this issue are outlined in section 3.4. Also, the tool is disconnected from the product and sprint backlog, meaning the tool only considers the total for each group of items.

Sprint marking as a Win or Loss is another way of tracking the success of a sprint and the results of the effort of the team. It is a coarse factor since there are only two distinct outcomes and minor steps may have huge or no impact on it. It is meant to alarm the team when it has glided off track, to take swift actions and bring the project back on track. The fine line between these two outcomes is although to be established. It is suggested by Downey and Sutherland [16] to mark the sprint as a Win if the focus factor is above 80% and the absolute value of a negative accuracy is not too large, which itself is yet to be determined. Otherwise the sprint is marked as a Loss. It is a failure to not finalize most of the sprint backlog items, but it is not a loss if a team finishes its duty earlier and
decides to adopt more work. A more complex formula could be introduced, depending on the projects and the environments.

An agile team functions well when it is small in size (around 5 members) where the communication path is simple and allows for information saturation. Teams that have an extreme size, such as 20 members or 1 member should be accounted for in the forecasts in the tool. Too large teams can cause problems in communication and work dispersion. Too small teams can burden the members in work and there will not be sufficient expertise in the team to tackle the challenges ahead. The tool shall warn the team in the planning phase of potential issues and perhaps suggest a lower amount of story points for the team.

It has been shown that a team does better if the interrupt pattern is implemented, regardless if interrupts are experienced [20]. Therefore, the interrupt lot should be somehow accounted for in the forecasts. One way would be to simply adopt less work than usual when anticipating an increasing number of interrupts. This also avoids having even more input fields. Having too many control knobs and parameters may harm the team, as the tool complexity increases and additional parameters will have to be considered every time.

When creating a sprint in the tool, the users specifies the start and end dates of the sprint. The tool can then compute the duration of the sprint. Studies show that when determining the sprint length, two major factors should be considered: the variety and the complexity of the product backlog items [27]. As the tool is disconnected from the backlogs, the user could be warned of potential problems, as described in section 3.4. The suggested forecast could be adjusted, to account for the issues that might arise, when the sprint length is of extreme size, e.g. 16 weeks or 2 days.

Currently, the tool allows forecasting of the effort in hours to be applied into the sprint. It is never accurate to the minute that the user will spend the number of forecasted hours. Users might have extra meetings, be tasked with other duties, have sick leaves and be away for many other reasons. Enabling the user to input actual effort applied, in hours or the prevailing usage of time unit, is essential if there are large discrepancies between forecasts and results. The input fields could be pre-filled with the time forecasts to avoid time waste in non-productive work. The sprint outcome would then better correlate with the sprint input. One factor that would immediately become more accurate is the velocity factor, which is a ratio of velocity to a time unit.

The tool suggests a fixed-point forecast of the number of story points to be enacted. There has been demand by the test users for a wider band of suggestions by the tool. One way to present it is as an interval with minimum and maximum values that would perhaps correspond to the worst and best velocity factors achieved over a period that stretches a pre-defined number of sprints in the past. The fixed point previously suggested would be the prime point on the interval.
The tool interface that the user interacts with could also be improved. Many users in the test group found the interface somewhat complicated to use or not quite intuitive. It has been suggested to reform the flow of the planning session in the tool. One suggestion is to make the flow more wizard-like where the user is guided through the various stages of the planning process. Another issue is that users forget to input the data into the tool at later stages. Data that is not inserted right away could be lost later. One solution to mitigate that is to send out a reminder, e.g. through e-mail or SMS, a number of days before the end of the sprint.

4.3 Threats to validity

The time constraint has made it rather difficult to produce statistical significance in the measurements. Agile teams often have iterations of several days or weeks and that is a very long-time span for a master’s thesis, thus more time is desired to study the consequences of the usage of the tool. The teams themselves acknowledged that the allotted time before evaluation was very short in respect to the nature of the software development activity. It remains unclear the extent to which related activities such as budgeting and release planning will be accurate.

The usability of the results presented here might not be applicable to all agile teams. That is due to the small tailored set of metrics towards Scrum teams. A revision may be required to accommodate other agile methodologies with a wider range of metrics that are used by the community.

The users in this project are in an organization that had not so long ago adopted the agile framework, hence the users might not have gathered enough experience in agile development. Also, the organization is in the public sector where some bureaucracy might have played a role in the organization of the agile teams. More time should be spent studying private companies that operate mainly in the software industry who are well versed with the agile principles.

Only one user group was put into study. Different groups do not necessary have the same background and especially in an agile environment where the rules and requirements are lax. They might have different processes to accomplish their work. A distinct perspective on the tool might have been procured.

4.4 Future work

The tool could be developed further to support the proposed enhancements that were discussed earlier. More evaluations may be required to establish legitimacy though, as the suggestions are derived from theoretical studies. Certain thresholds and limits are to be determined for some of the enhancements to be properly utilized.
The in-depth study was by no means to be comprehensive covering all factors and aspects that come into play in the agile work. The purpose was merely to support and give more depth to the tool and pave a way for scrum teams to achieve higher planning accuracy. More research could be conducted to uncover other factors that play a significant role in the progress of a scrum team and investigate the causes for deviations from plan, such as in budgeting and release planning.

The access to the data in the tool might have to be restricted due to security reasons. The foundation for a role-based access has been laid. The tool should support access based on roles and restrict viewing of data across teams or groups of users.
5 Bibliography


6 Appendix

6.1 Survey

*Scrum Team*
1. Sprint length in weeks
   - 1-2
   - 3-4
   - 5-6
   - 7+
2. Estimation unit for backlog items
   - Story Points
   - Ideal hours/days
3. If story points, what number series do you use?
   - Fibonacci
   - Power of 2
   - T-shirt sizes (Small to X-large)
   - One size only
4. Your technical experience level
   - Novice
   - Competent
   - Professional
5. Duration of Agile work (whole organization)
   - 1-6 months
   - 6-12 months
   - 1-2 years
   - 2-3 years
   - 3+ years
   - I don’t know
6. Duration of Agile work (your team)
   - 1-3 months
   - 3-6 months
   - 6-12 months
   - 1+ year
   - I don’t know

*Sprint planning*
7. Do you find it challenging to accurately plan sprints?
   - Yes
   - No
8. If yes, what do you find problematic with sprint planning? (Several options may be marked)
   - Estimating size of product backlog items
   - Estimating team capacity for a sprint
9. Are you familiar with the INVEST criteria for sprint backlog items?
   - Yes
   - No

10. Is User Story Mapping used within your team?
    - Yes
    - No

11. Mark the following you believe you often experience (Several options may be marked)
    - Team problems (team work, changing members, communication, work distribution, commitment etc.)
    - Managerial problems (Organization, Product Owner, Scrum Master, Linear organization)
    - Significant budget overruns
    - Missing release date
    - Spending time fixing bugs
    - Doing administrative work rather than providing direct value for the customer
    - Too much in progress in a sprint
    - Too many team interruptions (new requests from stakeholders)
    - Discrepancies between capacity forecast and actual work
    - Other

Planning Tool (Rate 1 to 5)
12. Get an estimated amount of work (story points/ideal days etc) per sprint
13. Visualization of velocity
14. Visualization of forecast accuracy
15. Visualization of team capacity
16. (Optional) Other functions or metrics you would like to see

17. (Optional) How Agile do you believe your team is? Describe what you think is missing.
6.2 Interview session

The interview was conducted on 7th June 2017. Each bullet point under each question is a response from one of the representatives. The order of the responses from the respondents is the same for all questions.

1. Did the tool hinder the ordinary planning process in any way?
   - The tool has not obstructed the planning work in any way. We have adopted many tools in our agile work. Although the tool is easy to use, it should be complemented with agile coaching to better utilize the value of the tool.
   - Not at all. The only issue that I can think of is that the team must remember to input the values at the different sprint events.
   - Not really. The only issue we have experienced is that the tool is not flexible enough to consider many factors we consider important, although unrelated to the planning phase. Once, when the pre-requisites for the agile work and structure had changed, the tool could not be utilized as intended. It could then not be used and extra care had to be taken when using the tool.

2. Did you experience difficulties understanding how the tool works?
   - Yes, a little bit in the beginning. As the tool is web-based, it is not fully compatible with all popular web browsers; not all labels and fields were positioned properly or fully visible. We as a state agency have policies on which browsers and free tools we can use. The tool also had some issues synchronizing the different view components. It was difficult to understand the planning process in the tool; the process flow was not intuitive.
   - No, it was straightforward. It just needed a trial round to get the hang of it. The terms used at some places were odd though.
   - No, not at all. Just some instructions were enough as the process flow is self-explaining and seems to be logical.

3. Do you believe the tool is breaking the Agile Manifesto where it is explicitly stated “Individuals and interactions over processes and tools”?
   - No, we do not believe so. An agile team is supposed to adopt tools the team deems fit; the team can be in full compliance with the agile manifesto but still pick what is best for the team.
   - No. The team members are very familiar with each other. We work as a unit and have no dependency on the tool. We merely have a glance over the figures the tool displays, rather than thoroughly complying with whatever it suggests. Therefore, interactions among team members is preferred over following a process or a tool. Moreover, we have a very influential person in our team, which strengthens our compliance with the manifesto even more.
   - No, every team must have tools to aid in its endeavors. It is even stated that waste should be minimized. Although, special care should be taken when carrying out extra administrative work that does not provide direct value and common sense should be applied.
4. Did a suggested sprint forecast help plan your sprint?
   - Yes, it has helped, although not much but it has been regarded as a hint. We have taken the suggested forecast into consideration along with many other factors. We applied judgement on a per case basis and what sounds reasonable.
   - Yes, it has helped us settle the problem of adopting too much work to handle at the beginning of a sprint. We consider the suggested forecast as a plausibility factor.
   - Yes, it has helped rather as a verification of the figures we come up with on our own. The numbers are somewhat abstract, so we usually first discuss the work that we are to commit to and then utilize the tool to verify that we are on track. I would rate the tool with 4 out 5, that it helps agile teams in the planning phase.

5. Did you experience a higher degree of accuracy between the projected amount of work for a sprint and the outcome?
   - It was a tad difficult to evaluate. We had large fluctuations in the results from time to time. Largely due to unforeseen events.
   - We had some vague definitions of user stories and sub-tasks, therefore we were forced to pad the estimates when we felt uncertain of the extent of the stories. There have also been variations in the uncertainty of estimations, therefore we could not directly conclude whether the accuracy has increased.
   - We had been using similar measures as in the tool, so the results do not signify an improvement. But we still believe in the model and think of it as a good measure to increase the accuracy.

6. Has the quality of the code base improved because of the usage of the tool?
   - It is difficult to assess in a such short period of usage. However, the scrum framework incorporates quality revision of our processes and consequently, the software quality is impacted.
   - It is difficult to assess in a such short period of usage and we are in the process of improving our estimations and many other factors. It would take us many more sprints as quality is quite an abstract term.
   - Yes, I believe so. It is hard though to deal with the bugs that we are experiencing, so we cannot tell for certain that we will attain a higher quality.

7. Has the tool helped your team make more accurate release and budget plans?
   - We believe that it is going to help in the long run. Especially when visualizing moving averages of some of the parameters.
   - When it comes to release planning, it seems that things have become more agile that we are distributing work more effectively, hence more accurate plans. We also believe that more accurate release and budget plans can be made in the long run.
When working in an agile environment, things tend to become very accurate, at least in our team. It is a tad difficult to forecast far into the future but with the parameters displayed, one can draw a curve and forecast with small error margins. We believe budgeting can also become more accurate, following the release plans.

8. How has the visualization of the forecast accuracy helped your team?
   - It has helped us a lot to see trends. We also judge ourselves if we can take on more work based on past results.
   - We have only peeked at those figures but we believe that it would help an agile team.
   - It is pleasant to see trends and tendencies. It feels like it helps a lot, as a verification that we are thinking right.

9. What other factors do you believe are vital to be considered in a planning tool?
   - We usually compute the amount of extra work that is to be done, that is the processing of the new requests. We call it the focus factor, which we believe should be incorporated in the tool.
   - The suggested forecast should be rather an interval than a fixed point. The tool takes in forecasted number of hours for the sprint but there may be need to input actual number of hours that has been put into the sprint.
   - The tool might have to reiterate quickly over the planning phase, as things might have been missed. We usually do that manually in our team. The focus factor should be incorporated we believe. The tool might have to aid the user in understanding the figures and reason about the results. The tool should not have too many functions, only the ones of vital importance.