UNIVERSITY OF OSLO
Department of Informatics

# Empirical research on relative and absolute effort estimation in software development projects

Master thesis
60 credits

Ivar Fredriksen

**November 2, 2009**

UNIVERSITAS OSLOENSIS • MDCCCXI •

# Summary

Inaccurate estimates of software development effort, is a recurring problem, causing budget and schedule overruns in software development projects. Therefore an improvement of estimation methods would be beneficial for the software development process.

Relative estimation methods, were effort estimates are made by looking at the relative difference in effort, as opposed to absolute estimation methods where effort is estimated directly in for example work-days, are becoming increasingly popular.

The main focus of this thesis is to investigate if using a relative estimation method for software development effort estimates has a positive impact compared to a similar absolute estimation method that does not compare effort relatively.

In agile software development, two methods for effort estimation are common. The first is a relative estimation method which uses the relative unit *story points*, the other is an absolute estimation method that uses the absolute unit *ideal days*, which are work-days if no interruptions could be assumed.

Both story points and ideal days are used to estimate effort, but the process used to arrive at the estimates differs. When estimating with story points, a process focusing on the relative differences of required effort is used. Expected duration of estimated effort with story points has to be derived using historical data on performance.

Ideal day estimates are related to the effort of work-days, but are assumed to be easier to estimate than most likely effort in work-days, since interference does not have to be taken into account. Since ideal days do not equal calendar days, expected duration has to be derived for estimated effort in ideal days, like for story points.

To contribute to the task of improving estimation methods, a study compar-

ing consistency [1] of story point estimates with ideal hour [2] estimates, was conducted on a software development project.

A higher consistency will enable a more reliable conversion from estimates in story points or ideal days to calendar time.

The studied project consisted of a team of summer interns in a Norwegian software development company. They worked using the scrum methodology with four sprints over four weeks, alternating every week between estimating with story points and ideal hours. A second team of employees from the same company supplied additional effort estimates, to study the effect of estimating the work of others.

An analysis on collected empirical data points in the direction that using a relative effort estimation method gives more consistent estimates compared to an absolute effort estimation method. There was higher consistency for story point estimates compared to consistency for ideal hour estimates. This suggests that using a relative estimation method has a positive impact on software effort estimation compared to a similar method that does not focus on comparing effort relatively.

The data also suggest that using a relative estimation method improved estimation performance when estimating the work of others, compared to when an absolute estimation method was used.

Since the dataset of the collected empirical data was small and only one project has been studied, the findings must be interpreted carefully. Further research on relative estimates should gather more data from more than one project. This could result in more robust statistical analyses than those presented in this thesis.

[1] In this thesis high consistency is viewed as small variation in the ratio for actual effort/estimated unit (story points or ideal hours)

[2] Essentially the same as ideal days, using hours as base instead of days

# Acknowledgements

I would like to thank my advisor Magne Jørgensen for sharing his insight, providing lots of valuable feedback for the thesis, and for many interesting discussions both related and unrelated to the research.

A special thanks goes to Mike Cohn for inspiring feedback and for inviting me to attend his scrum courses in California!

Thanks to Nils Christian Haugen for sharing raw data from his research and for inspiration on the thesis topic.

Last but not least, thanks to all the folks at Iterate and the summer interns who participated in the study project.

# Contents

# 1 Introduction

## 1.1 Process improvement

In 1911 F.W. Taylor wrote the book: "The principles of scientific management" [17]. Here he describes how decisions based upon traditional rules of thumb should be replaced by an adjusted process after carefully studying the people doing the work and identifying where there is room for improvement. The method of observing, methodically recording values, and adjusting a process based on these observations was new to the world, and with this Taylor pioneered process improvement.

Deming later builds on Taylors principles in his widely used Plan Do Act Check model (PDAC)[2], leaving Taylors principle that there is always a "best way of doing something" in favor of continuous improvement.

## 1.2 The estimating problem

One of the largest problems the software development industry faces is the notorious tendency for exceeding the original estimates [18]. A review of surveys by Moløkken and Jørgensen shows that average cost overrun in software development projects is in the range 30-40% [14]. The 1994 Chaos report by the Standish group suggest that average cost overrun is as high as 189% [5]. There is however reason to doubt the validity of the results in this report [9].

## 1.3 Agile estimating

In February 2001 a group of 17 process experts gathered in Utah to discuss common ground. Especially how processes in software development could be improved and promoted. The result of this meeting was the "Manifesto for Agile Software Development" [3]. The four central values of this manifesto are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

> That is, while there is value in the items on
> the right, we value the items on the left more.

The cornerstones of agile software development, iterative, evolutionary and incremental software development, are viewed by some as the "modern" replacement for the well-known waterfall model [11]. However both the published and practices roots of iterative and incremental development (IID) go back decades [11].

While agile software development might not be as novel as it appears at first glance, the agile community has established IID as common practice in a large part of the software development industry. This can be seen as a step in positive direction; research indicates that IID projects are more likely to have less effort overruns than comparable waterfall projects [15].

In agile projects functionality is often expressed as user stories. A user story describes functionality in the everyday language of the user. There are two common ways of estimating user stories described in Cohn's book "Agile Estimating and Planning" [1]: Story points and ideal days.

Story points are unit-less measures which is used to estimate the effort involved in user stories relative to each other.[3]

Ideal days are the effort in work-days if it could be assumed: (1) The user story that is being estimated is the only thing you will work on. (2) You

[3] Some ambiguity exists for the definition of story points, see section 2.5.1 for more details on how story points are interpreted in this thesis

have everything at hand that is needed to solve the user story, and (3) There are no interruptions [1].

Both story points and ideal days are used to estimate effort, but the process of arriving at the estimates differs. When estimating with story points, a process focusing on the relative differences in effort by relating the story point values of user stories to each other is used. When estimating with ideal days, user stories are given ideal day estimates that are related to the effort of work-days.

To be useful for planning, the duration of user stories estimated in story points or ideal days must be derived. This is achieved by recording the number of story points or ideal days completed over a fixed period of time. This number is called velocity [1], and can be used to calculate how much longer a team needs to work to finish remaining user stories in a project. It is possible to derive duration of user stories without historical data by forecasting velocity. However that is beyond the scope of this thesis and will not be discussed here.

## 1.4 Relative judgment

In 1956 Miller writes in his famous article "The magical number seven, plus or minus two(...)":

> Let me summarize the situation in this way. There is a clear and definite limit to the accuracy with which we can identify absolutely the magnitude of a unidimensional stimulus variable. I would propose to call this limit the span of absolute judgment, and I maintain that for unidimensional judgments this span is usually somewhere in the neighborhood of seven. We are not completely at the mercy of this limited span, however, because we have a variety of techniques for getting around it and increasing the accuracy of our judgments. The three most important

> of these devices are ( a ) to make relative rather than absolute judgments; or, if that is not possible, ( b ) to increase the number of dimensions along which the stimuli can differ; or ( c ) to arrange the task in such a way that we make a sequence of several absolute judgments in a row. [13]

Relative judgment is one of the oldest subjects from experimental psychology, both Laming [10] and Helson suggest [8] that psychophysical judgment is relative.

When deriving duration of user stories from effort estimates, it seems plausible that more accurate results could be achieved if these effort estimates were the result of using a relative estimation method rather than an absolute estimation method.

There is a huge leap from relative judgment on one-dimensional stimulus variables in a psychophysical setting to relatively comparing effort required for user stories in software development. There is no previous research on this subject in software development. Therefore it is interesting to study the effect of using relative effort estimation methods, and if our ability to do simple relative judgments more efficiently than absolute judgments, holds true at this much higher grade of complexity.

Comparing relative effort estimates using story points, to absolute effort estimates using ideal hours, is the main topic for this thesis. With this work I hope to contribute to the ongoing process of improving software effort estimates in software development.

# 2 Terms and measures

In this section terms and measures used in the thesis and the studied project will be presented.

## 2.1 Employees and Interns

- Employee in the company Iterate with domain knowledge of the project used in the study.
- Intern working at Iterate the summer 2009 with the project used in the study.

Quotes from employees and interns will be used throughout the thesis to support arguments or to illustrate points, and will be on the form 'E-(number)' and 'I-(number)' respectively. If the moderator of the estimation sessions is quoted, this is denoted by 'M'.

## 2.2 User Story

A user story describes a feature in the system in the everyday language of the user at a high level of abstraction.[4] User stories are typically written with the template:

As a/an <user role> <I want to...>, <so that...>

Examples:

As a player, I want to start a new game, so that I can play.

As an administrator I want to kick cheating players from games, so that the other players can enjoy a fair game.

[4] `http://en.wikipedia.org/wiki/User_story`, 18.09.2009

## 2.3 Scrum

Scrum is a software development framework that is commonly used in agile software development first introduced by Ken Schwaber [16]. The principles of scrum can be used outside of software development. [5]

When working with scrum all development happens in sprints or iterations, a sprint is a set length, normally 1 to 4 weeks, for when the team must deliver potentially shippable software. Potentially shippable software means; features that are done, tested, working well, but might not give business value. Sprints will always finish on time, they are time boxed so that scope can be dropped but deadline is fixed. Features, in the form of user stories, that are planned for the project are put in a product backlog (list of undone work).

In scrum there are four kinds of meetings that are central to the process:

- Daily standup meeting: Each day, before work begins, team members gather to answer three questions:

    1. What did you do yesterday?
    2. What are you going to do today?
    3. Are you facing any obstacles that needs to be removed?

- Release planning meeting: the release is planned and user stories from the backlog is roughly placed in available sprints.

- Sprint planning meeting: user stories are put into a sprint from the backlog.

- Sprint review meeting: the team presents or demos what they have done last sprint.

[5] http://en.wikipedia.org/wiki/Scrum_(development), 19.09.2009

The product backlog is estimated during a release planning meeting. In scrum, teams are self organizing. This means that within a sprint, the team decides for itself how it will go about doing the work.

## 2.4 Planning poker

Planning poker is a consensus based estimating game to estimate user stories. First described by Grenning in 2002 [4] and later popularized by Cohn [1].

The most recent version of planning poker uses a non linear scale which is somewhat similar to the Fibonacci scale: $0, \frac{1}{2}, 1, 2, 3, 5, 8, 13, 20, 40, 100$. The argument for this, as opposed to use a linear scale, is that as size and complexity grows, so does uncertainty. Thus if a developer wants to play a 10, he is forced to reconsider if the uncertainty warrants a jump to 13 or accept an estimate of 8.

These are the steps of a planning poker round used in this study:

1. A user story is read aloud by a moderator, then there is a discussion about the user story.
2. Everyone selects a card, and place it face down on the table in front of them.
3. Cards are flipped simultaneously.
4. The estimators of the low and high cards explain their thoughts on the estimate.
5. A brief discussion if anyone else want to explain their viewpoints.
6. Repeat the steps until consensus is reached by everyone selecting the same card, or by agreeing after discussion.

If a team openly discusses estimates, there is a chance of anchoring effects. For example, if an estimator says something like: "I think this user story is very small", the statement unconsciously 'anchors' itself in the minds of the other estimators. There is a good chance that another estimator who originally thought the user story was large, adjusts his view without ever discussing any initial thoughts prior to the 'anchoring'.

The idea is that planning poker will help reduce anchoring effects, because cards are revealed simultaneously, as well as making estimates less optimistic and more accurate than statistical combining of individual estimates [20] [6].

## 2.5 Relative estimates

For this thesis, when referring to relative estimates or relative estimation methods it means: user stories that are estimated relative to each other using story points. When referring to relative sprints, it means sprints that are estimated with story points.

### 2.5.1 Story points

Story points as a unit for estimation has become quite popular in the agile community recently. However, the literature concerning story points is somewhat vague, for example, Cohn writes:

> The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it, and so on. [1]

When it later in the same book is said that: "Story points are purely an estimate of the size of the work to be performed"[1] one is left to wonder what that might mean. This might be linguistic cantankerousness, but a more precise definition or a less ambiguous explanation of story points would have been very welcome.

In this thesis story points are interpreted as *a relative unit-less measure of effort*. This way the good idea of keeping story points as a pure measure of effort is intact. In addition story point values remain abstract by explicitly branding the measure as unit-less. Therefore there are no reference points for story points, except the story point values of other user stories. To my understanding, this is the essence of story points.

A quick search on the Internet illustrates the current general ambiguity of story points in the community that use them:

“Story point is a random measure used by Scrum teams. This is used to measure the effort required to implement a story. In simple terms its a number that tells the team how hard the story is.”[6]

“Story points (referred to as just points in this article) are units of relative size used in estimating software requirements as an alternative to units of time. Points are a measurement of complexity and/or size of a requirement as compared to the duration to complete that requirement.”[7]

“Story Points are a relative measure of feature difficulty/complexity that are useful for planning. If one feature takes twice as much effort as another, it will have twice the story points.”[8]

“Story points are just relative measures of the effort involved in estimating a user story (...) Story points are a pure measure of effort.”[9].

### 2.5.2 Estimating with story points

To get started with relative estimates the team must agree on a baseline to have a reference point for the rest of the estimates. There are two common ways to get started [1].

[6] `http://agilefaq.net/2007/11/13/what-is-a-story-point`, 28.09.09

[7] `http://en.wikipedia.org/wiki/Story_points`, 28.09.2009

[8] `http://www.agilegamedevelopment.com/2006/03/hours-vs-story-points.html`, 28.09.2009

[9] `http://edgibbs.com/2006/03/11/story-points-estimating/`, 28.09.2009

1. Select a user story that is expected to be one of the smallest, and then assign it one story point.

2. Select a user story that is assumed to be about medium sized compared to the others, and give it a story point value somewhere in the middle of the range that will be used.

Once the baseline is set, the team starts estimating user stories giving them an estimate in story points relative to the baseline.

It is important to note that as more user stories are estimated, the new estimates also are considered to be reference points in addition to the baseline.

## 2.6 Absolute estimates

For this thesis, when referring to absolute estimates or absolute estimation methods it means: estimates of the effort of user stories using ideal hours. When referring to absolute sprints, it means sprints that are estimated with ideal hours.

### 2.6.1 Ideal time

Ideal time is an estimate of how long it will take to complete a user story in an ideal setting. That is to say if it could be assumed: (1) The user story that is being estimated is the only thing you will work on. (2) You have everything at hand that is needed to solve the user story, and (3) There are no interruptions [1]. Ideal time is typically expressed as ideal days, or ideal hours, depending on the context.

As story points, ideal time is subject to some interpretation. One of the big problems with understanding ideal time is how 'ideal' one judge ideal time to be. Figure 1 shows a normal distribution of possible estimated time for some task. Most likely time is at the peak. Where should the ideal time be?

There seems to be no definition or rule for this, and each estimator must make up their own mind as to where they draw the line. Judging from the discussion done during the estimation sessions, it seems both teams are quite far to the left of this distribution:

> E-5: Once there is just some tiny problem with hibernate, all those 40 hours will be gone fixing it.
> E-1: Yes, but we must think in ideal time, we won't have any trouble with hibernate.
> E-5: I agree with that.
>
> —
>
> I-5: Oh! 20 ideal hours? That's like ... 3 weeks actual work? (short laugh)

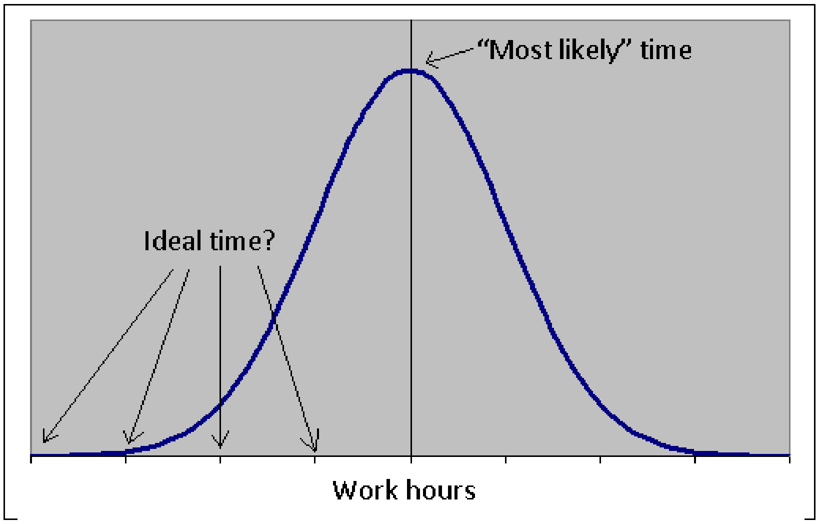


Figure 1: Where is ideal time?

### 2.6.2 Estimating with ideal hours

Getting started with ideal time estimates is less work than with relative estimates. No baseline need to be found, and user stories can be estimated at once. Estimators try to agree on an estimate for each user story in ideal hours.

## 2.7 Consistency

What is meant by consistency of estimates is not self-explanatory. In this section I will clarify what is meant by consistency in relation to estimates in this thesis.

- Let S be a set of user stories.
- Let X be a set of values for *actual effort in hours* for the user stories in S.
- Let Y be a set of values for *estimates* for the user stories in S.
- Let Z be a set of values that is the result of $\frac{X_i}{Y_i}$ for i = 1...n.

Consistency for estimates off the user stories in S is expressed in two ways:

1. The standard deviation of the values in Z.
2. The maximum value minus the minimum value of Z.

Estimates are given in either story points or ideal hours. The first approach gives a value that is representative for the whole set with the standard deviation, whilst the second approach gives an indication of how much the total spread the values have (max-min). For both approaches a high number means low consistency, while a low number means high consistency.

### 2.7.1 Why consistency and not accuracy?

To exemplify, imagine a project with six user stories. Each user story is given three different estimates in story points. This example could also have used ideal hours. The three sets of estimates for the project, actual effort in hours, and the ratio for actual effort/estimate for each user story (the Z-columns) is as shown in table 1.

Values for consistency can be seen in table 2.

To be useful for planning, expected duration of user stories must be derived from the estimates. A high consistency will enable higher accuracy when doing this.

Let us assume user story 1, 2 and 3 are completed, and estimates are provided for all six user stories. We want to find expected duration for completing user story 4, 5, and 6.

For each set of estimates, the sum of actual effort for user story 1, 2 and 3 is divided by the sum of story points. This gives a value for how many hours one story point means for each set. This value is then multiplied with the sum of story points estimated for user story 4, 5 and 6 which in turn is the expected duration of user story 4, 5 and 6 for each set.

We see in table 3 that the sets of estimates with high consistency, 1 and 3, come close to the actual effort of 38 hours, while the set with low consistency, set 2, misses the target completely.

| User story | Set 1 | Set 2 | Set 3 | Actual effort | Z - 1 | Z - 2 | Z - 3 |
|:-:|--:|--:|--:|:-:|--:|--:|--:|
| 1 | 20 | 20 | 40 | 38 | 1,90 | 1,90 | 0,95 |
| 2 | 8 | 5 | 13 | 16 | 2,00 | 3,20 | 1,23 |
| 3 | 20 | 13 | 40 | 40 | 2,00 | 3,08 | 1,00 |
| 4 | 2 | 13 | 5 | 4 | 2,00 | 0,31 | 0,80 |
| 5 | 12 | 8 | 20 | 22 | 1,83 | 2,75 | 1,10 |
| 6 | 5 | 40 | 13 | 12 | 2,40 | 0,30 | 0,92 |

Table 1: Example Data

|  | Z - 1 | Z - 2 | Z - 3 |
|---|---|---|---|
| St. dev | 0,20 | 1,33 | 0,15 |
| Max - min | 0,57 | 2,90 | 0,43 |

Table 2: Example Data - consistency

|  | Set 1 | Set 2 | Set 3 | Sum actual hours |
|---|---|---|---|---|
| User story 1, 2 and 3 |  |  |  |  |
| Sum completed estimates | 48,00 | 38,00 | 93,00 | 94,00 |
| Sum Actual / Sum estimated | 1,96 | 2,47 | 1,01 |  |
| User story 4, 5 and 6 |  |  |  |  |
| Sum remaining estimates | 19,00 | 61,00 | 38,00 | 38,00 |
| Expected duration in hours | 37,21 | 150,89 | 38,41 |  |

Table 3: Example Data - expected duration

# 3 Research topics and research questions

In this section the research topics and research questions will be presented.

## 3.1 Research topic 1: The effect of using relative estimation methods compared to absolute estimation methods in software effort estimation

There is a growing interest and usage of relative estimation methods, such as estimating user stories in story points [7]. This makes empirical research on possible effects of using relative estimation methods particularly interesting.

### 3.1.1 Research question 1.1: Will relative estimation methods be more consistent than absolute methods when estimating user stories?

## 3.2 Research topic 2: The effect of having two separate teams provide effort estimates on the same set of user stories

Lederer and Prasad suggests in their nine guidelines for better cost estimating that the initial estimating task should be done by the final developers [12]. The interns and the employees did separate effort estimates on the user stories in the studied project. The effect of this is explored in this research topic.

Research topic two is only based on analysis of collected empirical data. Thus findings on this topic cannot be considered to be as powerful as if they were assumed prior to the study.

### 3.2.1 Research question 2.1: How good is estimation performance by the two teams compared to each other, and estimation method?

# 4 The study and research method

In this section a description of the studied project and methods used for data gathering will be presented, as well as threats to validity of the results.

Empirical data for this thesis was collected from a study which was conducted on a software development project. The project took place at Iterate, a small Norwegian software consultancy company from June $15^{th}$ to July $24^{th}$ 2009.

## 4.1 Participants

The development team consisted of five summer interns in the age range 22-25 years, being in their 3d or 4th year of studies at the Norwegian University of Science and Technology (NTNU). A second team of six employees at Iterate also participated to supply additional estimates for the project; the employees are between 25-35 years, and have 1 to 10 years of experience in the software industry. All the employees did not have time to participate every estimation session, so the number of employees present each estimation session ranged from three at the least and five at the most.

## 4.2 The Project

The task was in house development, adding functionality to already existing internal software called LeanCast.

LeanCast is a platform for sending internal messages and is also used to organize reimbursement of expenses, like taxi fares. LeanCast was developed using a Java with Spring for dependency injection, Hibernate for persistence, and Apache Wicket as web framework.

LeanCast is made with a plug-in architecture, and is built and deployed using Apache Maven. In addition to a web interface, LeanCast supports messaging from both XMPP and SMS.

The task for the interns was to integrate the possibility to register work hours in LeanCast so that the old hour registration system could be phased out. The project was carried out using scrum with one week sprints.

To be able to get comparable data for relative and absolute estimates, both estimation methods had to be used in the project. Several approaches to achieve this were considered:

1. Estimating half of the backlog in ideal time, and half in story points at the release planning meeting.

2. Splitting the backlog in half and estimate the first half in ideal time at the release planning meeting. Then later estimate the rest of the backlog with story points about halfway into the project.

3. Alternate between estimating every other sprint with relative estimates and absolute estimates in the sprint planning meetings.

The first approach was not feasible because the entire backlog of user stories for the project was not finished when the project started. The learning curve of doing estimates for the interns also had to be taken into account, as none of them had previous estimating experience from other software development projects.

The second approach also had to be discarded, for the same reasons as the first.

The last option; alternating between the two estimation methods every other sprint, gave the flexibility needed to create user stories before each sprint started. The greatest concern with this option was that switching estimation method every sprint could be confusing for the participants, see section 4.6.3 under threats to validity.

Cohn discourages estimating in the sprint planning meetings. He recommends estimating in the release planning meeting to be able to prioritize the backlog, make high level forecasts and make trade-offs between scope and schedule [10].

These arguments are not so important in this setting, because the estimates were not used for planning, and estimating each sprint was necessary for the research.

## 4.3 Preparations

The interns were given introductory lessons in scrum, story points and ideal time as well as planning poker.

A persona is a brief description of a fictional person often used in the field of human-computer interaction among others [19].

As an attempt to set a shared reference for ideal time a persona was created described as "The perfect student" - this persona would have to research material, but would understand everything instantly, then implement code perfectly with no errors the first time. How many hour this persona would use to solve the user story were considered the number of ideal hours the user story should be estimated at.

## 4.4 Estimation sessions

The data from the studied project was collected during the estimation sessions. The estimation sessions were video recorded to be able to look at discussions later and get richer data. It also provided the possibility for greater depth of analysis and decision-making. In addition to the video, all estimates were recorded during the sessions.

The team of interns and the employees did their estimation sessions separate from each other, preventing them from influencing each other. The different

[10] `http://blog.mountaingoatsoftware.com/when-should-we-estimate-the-product-backlog`, 12.10.09

estimates were not made available to the teams during study.

In sprint 1 the interns did their estimates first followed by the employees. This order was reversed for the remainder of the sprints. The reason for this was the role as stakeholders for the product held by the employees. By estimating first the employees could review the user stories, come to a common understanding of them, and possibly edit them, before they were estimated by the interns. In turn this saved the project from a lot of confusion and miscommunication, as well as helping to clarify some issues, that for an internal software project might otherwise have gone unnoticed.

Each estimation session followed roughly the same pattern:

- User stories were written on index cards and given to the teams prior to each estimation session.
- A brief discussion of the user stories.
- Planning poker was played for each user story.

Sprint 2 deviated from this pattern as it was the first sprint using story points. Before the team could start playing rounds of planning poker they had to set a common baseline to use as a reference to make relative estimates.

Sprint 1 did not include a discussion of all the user stories prior to playing planning poker. For this reason there was some confusion. See corrections and notifications for collected data in Appendix C.

Planning poker was chosen for the estimation sessions because it is a suitable method of estimating both story points and ideal hours and it also involves the entire team in the estimation process. Using the same scale for two different ways of thinking (relative and absolute) might be challenging, yet it was judged necessary as an isolation strategy for the study of the effect of the estimation methods. Planning poker is also fairly easy to explain and execute. Therefore it would not be quite so much overhead for the participants with regards to techniques used when estimating, and the estimates could be done within a reasonable amount of time.

Ideal hours were used instead of the more common ideal days, because of the size of the project. Ideal days would be a too coarse measurement for the user stories with a sprint length of 1 week.

## 4.5 Deviation from plan

Being a project in the real world there were some deviations from the plan. The project was originally planned with six sprints, each one week. It soon became apparent that this approach was not possible. Some time got lost when training the interns, and including some changes in the schedule only four sprints actually got completed.

Both teams used the same estimation unit each sprint. In the first sprint ideal hours were used, in the second story points. The third sprint was back to ideal hours, and in the last sprint story points were used.

## 4.6 Threats to validity

The complexity involved in experimentally comparing relative and absolute estimation methods is great. As a consequence, there are a several threats to validity.

### 4.6.1 Small dataset

The foremost threat to the validity of any analysis made on the results is the small size of the dataset. For this reason the results must be interpreted carefully. As an attempt to accommodate for this, relevant quotes from the participants in the study are included to illustrate points and strengthen arguments throughout the thesis.

### 4.6.2 Variation in actual effort and estimation complexity for user stories

The user stories in the four sprints vary somewhat in effort to implement. Because this was not known prior to the study, nothing has been done to compensate for this. For example all the user stories in sprint 2 have a fairly low value for actual hours reported.

The first round of planning poker usually does not end with consensus. There are different interpretations and understandings of a user story which is expressed with the cards that are played.

As a measure of estimation complexity for a user story the following is used: the number of possible cards to play inside the range of cards actually played in the first round of planning poker. The range used was $0, \frac{1}{2}, 1, 2, 3, 5, 8, 13, 20, 40, 100$. A higher number for the range of possible cards is associated with higher estimation complexity. For example, if the cards 3, 5 and 8 are played, estimation complexity would be 3 for that user story. If the cards 2, 8 and 13 are played estimation complexity would be 5.

| | Employees<br>Number of user stories | | Interns<br>Number of user stories | |
|---|---|---|---|---|
| E-complexity | Story points | Ideal hours | Story points | Ideal hours |
| 1 | 3 | 0 | 2 | 0 |
| 2 | 1 | 4 | 2 | 2 |
| 3 | 2 | 2 | 1 | 3 |
| 4 | 0 | 1 | 1 | 3 |
| 5 | 0 | 0 | 0 | 1 |

Table 4: Estimation complexity for user stories

As is shown in table 4 the estimation complexity is higher for more of the user stories estimated with ideal hours.

This simplistic analysis is not meant to give an accurate expression of estimation complexity, but rather to have something tangible to compare the estimation complexity of user stories estimated with story points and ideal hours in this thesis. Values for estimation complexity overlap for the most part. Because of this estimation complexity is not judged to be the decisive factor for the results when comparing absolute and relative sprints in this thesis.

For the cards played each planning poker round see Appendix B.

### 4.6.3 Confusing for participants to switch estimation method each sprint

Changing the method used for estimating each sprint may have confused the participants, and perhaps affected some of the estimates one way or the other. Attempting to accommodate for this it was repeatedly stressed what kind of estimation method was in use every estimation session. For example, when using story points, the importance of seeing effort of the user stories in relation to each other, was emphasized.

### 4.6.4 The estimates were not used for planning

Since this was a very low risk internal project the estimates for this research were not used in any way for planning further sprints. This could matter for estimation performance, since there was no pressure to be accurate. However, the study was not to find an optimal way of doing software estimation, but rather to compare absolute and relative estimation methods. That the estimates were not used for planning is not assumed to have had a great impact on the results. The level of commitment can for example be assumed to be about the same for both estimation methods.

Participants did find the estimation sessions useful because they got to review and discuss the user stories in a more structured way than would have happened otherwise. This was especially true concerning the employees:

> E-3: The sessions helped with focus on what the interns were

doing, and what we really wanted from the project.

### 4.6.5 Hours reported by Interns

To correctly track the number of hours spent to solve a user story, is not an easy task. The interns were asked to note how many hours they spent on each user story, with an emphasis on being as close to reality as possible. The details on how they were to do this, was left to the interns, leaving the idea of a self organizing scrum team intact.

It is believed that the numbers reported by the interns were to the best of their ability correct, as they had no incentive, conscious or unconscious, to report wrong numbers.

# 5 Results

In this section data collected from the studied project and analysis on the data will be presented.

From the study there are two sprints each for relative and absolute estimates. The final sprint using story points only completes one user story. Because of this, calculating consistency for each sprint individually was not a good option. For readability of analysis, data from both sprints using story points are combined, and the same is done for sprints using ideal hours.

See Appendix A for user stories, and Appendix C for corrections and notifications on collected data.

## 5.1 Collected data from the studied project

Table 5 shows the estimates by employees and interns in ideal hours from sprint 1 and sprint 3, as well as actual hours reported for each user story.

Table 6 shows the estimates by employees and interns in story points from sprint 2 and sprint 4, as well as actual hours reported for each user story.

| User story nr. | Interns estimate | Employee estimate | Actual hours |
|---|---|---|---|
| 1.1 | 8 | 20 | 51 |
| 1.2 | 1 | 0 | 0,5 |
| 1.(4,6,7) | 36 | 40 | 96 |
| 3.1 | 13 | 40 | 39 |
| 3.2 | 13 | 40 | 54 |
| 3.3 | 5 | 13 | 102 |
| 3.4 | 3 | 5 | 1 |

Table 5: Ideal hour estimates and actual hours reported

| User story nr. | Interns estimate | Employee estimate | Actual hours |
|---|---|---|---|
| 2.1 | 8 | 3 | 5 |
| 2.2 | 13 | 5 | 5 |
| 2.3 | 5 | 13 | 15 |
| 2.4 | 5 | 13 | 15 |
| 2.5 | 5 | 13 | 15 |
| 2.6 | 3 | 3 | 7 |
| 4.2 | 13 | 40 | 37 |

Table 6: Story point estimates and actual hours reported

## 5.2 Notes on cross team comparison

One of the difficulties when comparing the estimates of several teams using story points is that each team comes up with some un-tangible agreement of what a story point should mean when they give the first user story a story point value as a baseline. This agreement becomes more apparent as more user stories are estimated relative to each other; this is especially true when using a consensus based estimation tool like planning poker.

This means that one story point of one team does not necessarily equal one story point of another team. Comparison between teams can still be achieved by using a conversion factor based on actual effort, as it is done in section 5.4.

The same problem is apparent for ideal time, possibly because the definition for ideal time also gives quite some room for interpretation. Even if no baseline is explicitly set for ideal time estimates, there must be some implicit agreement inside the team of how to interpret an ideal hour. Therefore it cannot be assumed that one ideal hour from one team is equal to one ideal hour of another team.

## 5.3 Consistency of estimation methods (RQ1.1)

RQ1.1: Will relative estimation methods be more consistent than absolute methods when estimating user stories?

To compare consistency between the absolute and relative estimates we need the values of estimated units divided on actual effort, se section 2.7 on consistency. In tables 7 and 8 the ratios actual hours / story points and actual hours / ideal hours are seen respectively. To be useful as a means to plan a sprint, these ratios should vary as little as possible.

| User story | Interns | Employees |
|---:|---:|---:|
| 2.1 | 0,63 | 1,67 |
| 2.2 | 0,38 | 1,00 |
| 2.3 | 3,00 | 1,15 |
| 2.4 | 3,00 | 1,15 |
| 2.5 | 3,00 | 1,15 |
| 2.6 | 2,33 | 2,33 |
| 4.2 | 2,85 | 0,93 |

Table 7: Relative estimates: ratios for actual hours / story points

| User story | Interns | Employees |
|---:|---:|---:|
| 1.1 | 6,38 | 2,55 |
| 1.2 | 0,50 | N/A |
| 1.4 + 6 + 7 | 2,67 | 2,40 |
| 3.1 | 3,00 | 0,98 |
| 3.2 | 4,15 | 1,35 |
| 3.3 | 20,40 | 7,85 |
| 3.4 | 0,33 | 0,20 |

Table 8: Absolute estimates: ratios for actual hours / ideal hours

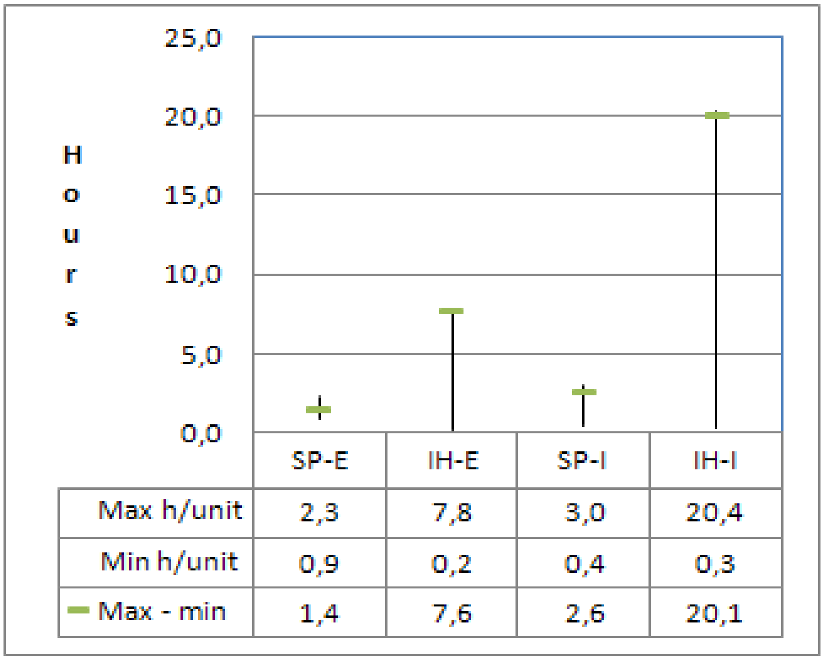
The maximum minus the minimum value from each of these four sets of ratios and the standard deviation of the same sets, give the values for consistency as shown in table 9. A lower score is better.

Figure 2 on the following page shows a graphical representation of hours pr. unit. A shorter line means higher consistency.

Values for consistency (max-min, st.dev) in the relative sprints were (2.62 , 1.16) for interns and (1.40, 0.50) for employees. These scores are significantly better than for the absolute sprints where the interns get consistency scores of (20.07, 6.96) and the employees have (7.65, 2.74).

| | Interns | | Employees | |
|---|---|---|---|---|
| | Story points | Ideal hours | Story points | Ideal hours |
| St.dev. | 1,16 | 6,96 | 0,50 | 2,74 |
| Max - min | 2,62 | 20,07 | 1,41 | 7,65 |

Table 9: Consistency (lower is better)



| | SP-E | IH-E | SP-I | IH-I |
|---|---|---|---|---|
| Max h/unit | 2,3 | 7,8 | 3,0 | 20,4 |
| Min h/unit | 0,9 | 0,2 | 0,4 | 0,3 |
| ▬ Max - min | 1,4 | 7,6 | 2,6 | 20,1 |

Figure 2: Hour pr unit distribution

Remember the definition of consistency in this thesis uses the maximum and minimum ratios for actual hours / estimated effort and the standard deviation the same ratios. The results are therefore very vulnerable to outliers.

Outliers are defined as values above $Q3+k(Q3-Q1)$ or below $Q1-k(Q3-Q1)$ for k = 3. [11] [12] Where Q1 and Q3 are the quartiles in the inter quartile range of the values for actual hours / estimated effort (found in tables 7 and 8).

[11] `http://en.wikipedia.org/wiki/Outlier`, 18.09.09

[12] `http://cc.uoregon.edu/cnews/spring2000/outliers.html`, 18.09.09

With this definition both values for user story 3.3 in table 8 were identified as outliers.

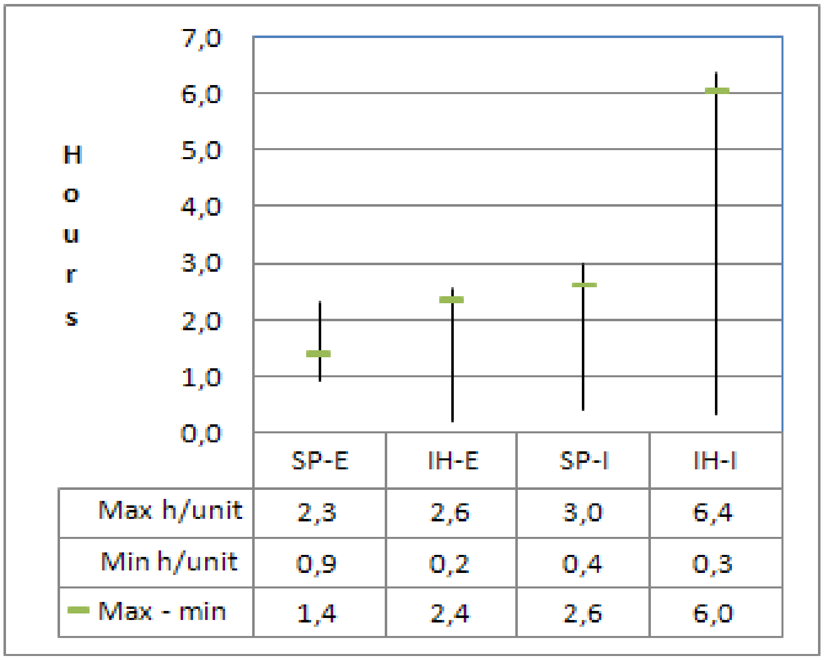
We now get the values of consistency as shown in table 10.

After removal of outliers, the difference in consistency is no longer as extreme. Relative sprints still have higher consistency with consistency scores of (2.62, 1.16) vs. (6.04, 2.28) for interns and (1.41, 0.50) vs. (2.35, 0.99) for employees.

Figure 3 shows a graphical representation of hour pr. unit, when the outliers are removed.

| | Interns | | Employees | |
|---|---|---|---|---|
| | Story points | Ideal hours | Story points | Ideal hours |
| St.dev. | 1,16 | 2,28 | 0,50 | 0,99 |
| Max - min | 2,62 | 6,04 | 1,41 | 2,35 |

Table 10: Consistency, outliers removed (lower is better)



| | SP-E | IH-E | SP-I | IH-I |
|---|---|---|---|---|
| Max h/unit | 2,3 | 2,6 | 3,0 | 6,4 |
| Min h/unit | 0,9 | 0,2 | 0,4 | 0,3 |
| ▬ Max - min | 1,4 | 2,4 | 2,6 | 6,0 |

Figure 3: Hour pr unit distribution, outliers removed

## 5.4 Comparing the teams on estimation performance and method (RQ2.1)

RQ2.1 How good is estimation performance by the two teams compared to each other, and estimation method?

To answer RQ2.1 the problem of comparing team effort with story points and ideal time as outlined earlier must be considered.

To accommodate for this, the following conversion factor is introduced to calculations in this section, the outliers identified in the previous section are not included in the analysis:

$$\text{Conversion factor} = \frac{\sum \text{Actual hours}}{\sum \text{Estimated unit}} \tag{1}$$

The values for the conversion factors can be seen in table  11.

| Unit | Team | Hours |
|---|---|---|
| Story point | Employees | 1.10 |
| Story point | Interns | 1.90 |
| Ideal hour | Employees | 1.67 |
| Ideal hour | Interns | 3.26 |

Table 11: Conversion factor for estimated effort in hours

To compute Relative error (RE) Magnitude of relative error (MRE) and absolute error (AE) used in the analysis, the following formulae are used:

$$RE = \frac{\text{actual effort } - \text{ (estimated effort } \times \text{ conversion factor)}}{\text{actual effort}} \tag{2}$$

$$MRE = \frac{\mid \text{actual effort } - \text{ (estimated effort } \times \text{ conversion factor)} \mid}{\text{actual effort}} \tag{3}$$

$$AE = \ |\, \text{actual effort} \ - \ (\text{estimated effort} \ \times \ \text{conversion factor})\, | \qquad (4)$$

The conversion factors seen in table 11 are expressions of how the estimated units would translate to hours if the actual hours used to solve the user stories were known. Therefore the values for MRE, RE and AE reflect a higher degree of realism than is actually the case. However we are not trying to indicate the degree of realism, but rather the difference in estimation performance between the teams. Since the modification is done to both teams and both estimation methods, the results are considered to be meaningful for comparing estimation performance between teams and estimation methods.

Values for RE and MRE can be seen in table 12, values for AE can be seen in table 13.

| | | | Mean | | Median | |
|---|---|---|---|---|---|---|
| Unit | Team | N | RE | MRE | RE | MRE |
| Ideal hours | Employees | 6 | -1.10 | 1.65 | 0.04 | 0.53 |
| Story points | Employees | 7 | 0.10 | 0.19 | 0.05 | 0.10 |
| Ideal hours | Interns | 6 | -2.32 | 2.56 | -0.16 | 0.36 |
| Story points | Interns | 7 | -0.63 | 1.09 | 0.33 | 0.37 |

Table 12: Relative error and Magnitude of Relative Error

| Unit | Team | N | Mean AE | Median AE |
|---|---|---|---|---|
| Story points | Employees | 7 | 2.14 | 0.70 |
| Story points | Interns | 7 | 8.57 | 5.48 |
| Ideal hours | Employees | 6 | 15.86 | 15.16 |
| Ideal hours | Interns | 6 | 12.16 | 10.18 |

Table 13: Absolute estimation error in work-hours

The graphs at the end of this section (figure 4 to 7) visualize the relationship between actual hours and the estimation methods for the teams. Regression lines are drawn through origo to give an indication of where the teams have over and underestimated. A dot over the line is an underestimate and a dot under the line is an overestimate. Distance from the line indicates severity of error.

Graphs of sprints using story points are found in figure 4 (employees) and figure 5 (interns).

Graphs of sprints using ideal hours are found in figure 6 (employees) and figure 7 (interns).

### 5.4.1 Team comparison for sprints using absolute estimation

We see that the interns had marginally better estimation performance than the employees in the sprints using ideal hours judging by the common case, with a median MRE of 0.36 and 0.53 respectively.

However the mean MRE for the interns is much larger than for the employees 2.56 vs. 1.65, suggesting the interns had larger errors in extreme cases.

By looking at the graphs on the following pages it is evident that, both interns and employees underestimate some user stories and overestimate others. Negative mean RE for both interns and employees suggest a tendency for overestimates, -2.32 for interns and -1.10 for employees.

### 5.4.2 Team comparison for sprints using relative estimation

The employees have better estimation performance in the relative sprint compared to the interns in the common case, with a median MRE of 0.10 vs. 0.37. Further, a mean MRE score of 0.19 for the employees suggest that there were no large errors, while this is not the case for the interns who had a mean MRE of 1.09.

The interns have a slight tendency for overestimating, with a mean RE of -0.63. The employees have no strong bias with a mean RE of 0.10.

### 5.4.3 Method comparison

In the sprints using ideal hours, there is more estimation error judging from mean AE values of 12.16 for interns and 15.86 for employees, compared to 8.47 and 2.14 in the sprints using story points.

To sum up, both teams have quite poor estimation performance when using ideal hours, the interns do not improve much when using story points, while the employees has very good estimation performance when using story points.

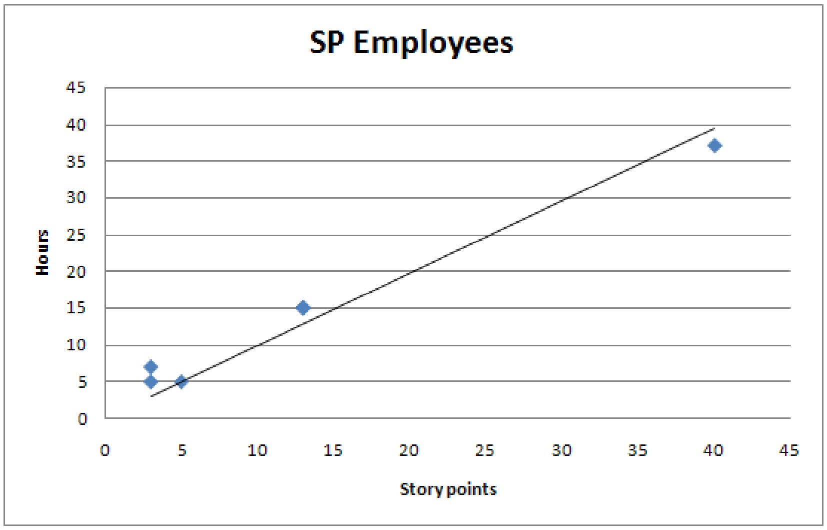


Figure 4: Story points vs. work hours for Employees

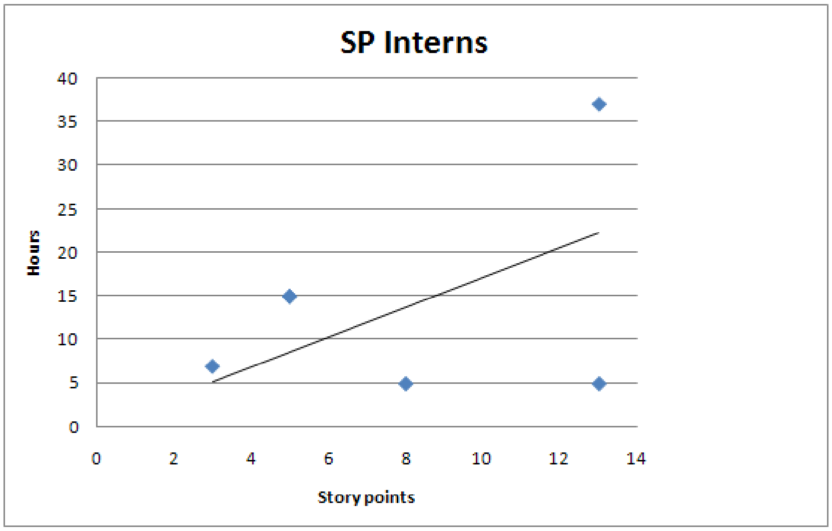


Figure 5: Story points vs. work hours for Interns

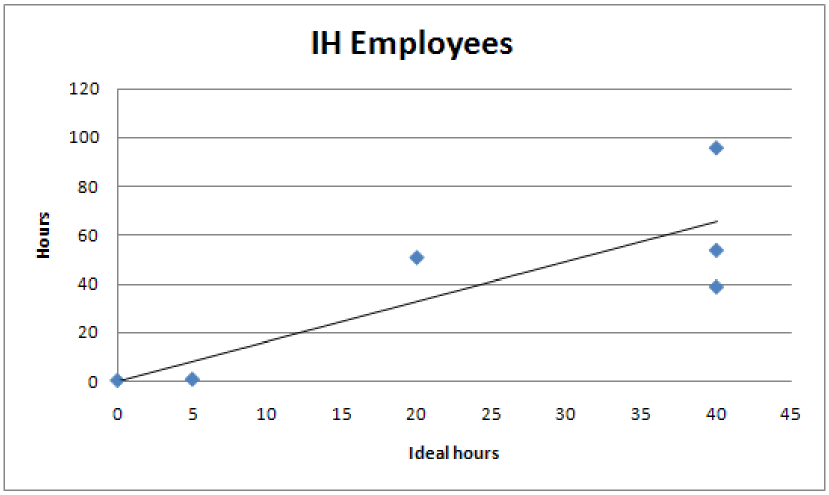


Figure 6: Ideal Hours vs. work hours for Employees

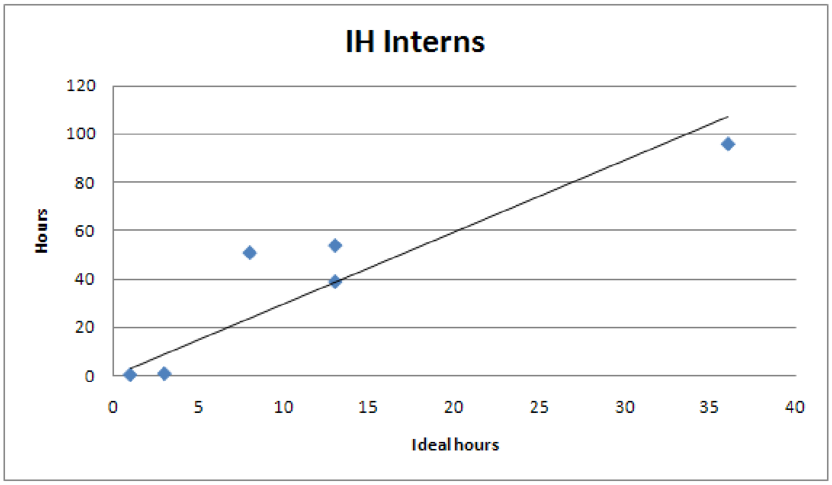


Figure 7: Ideal Hours vs. work hours for Interns

# 6 Discussion

This section contains a discussion around the results and the thesis in general.

## 6.1 Are relative estimates more consistent? (RQ1.1)

The analysis shows that the relative estimates were more consistent for both teams. With outliers removed, relative sprints were much more consistent than absolute sprints. A higher consistency for relative estimates compared to absolute estimates was expected. However, extra caution must be taken when interpreting the results, because of some difference in estimation complexity between the user stories in the relative and the absolute sprints.

For dealing with the problem of variation in user story estimation complexity, a better option might be to compare the standard deviation for actual hours of all user stories estimated at the same number of story points or ideal days/hours. This could have been used as an alternative measure for consistency, if more data were present.

## 6.2 Estimation performance and method comparison (RQ2.1)

When comparing estimation performance between the teams in the absolute sprints, not much difference was found, estimation performance for both teams was poor.

In the relative sprints, estimation performance by the employees is better than the interns, and also much better than in the absolute sprints. This suggests that using a relative estimation method makes it easier to estimate the work of others, compared to an absolute estimation method.

Research by Haugen [6] suggest:

- Planning poker can be less accurate when there is no experience from similar tasks.

- Planning poker can make errors larger in extreme cases.

The poor estimation performance by the interns supports the first case, while the large errors made by both teams in the absolute sprints supports the second case.

Two additional factors were assumed to have a positive impact on the estimation performance of the employees:

- The employees had more domain knowledge of their own system than the interns.
- Two of the employees who participated in the study worked on the same software last year (2008), when they themselves participated in a similar intern program at the same company. This closeness in experience-level might have helped the employees in questions to relate the position of the interns more easily than more senior employee's might.

## 6.3 Finding the baseline

Finding a baseline is probably the hardest part of using story points. It can feel intimidating to pick a value, which before it is seen in relation to story point values of other user stories, does not mean anything. Before the baseline can be set, estimators need to discuss the user stories to be estimated, to get some idea of the relative difference in effort involved in solving them. The interns and employees had some differences in the way they carried out this discussion.

### 6.3.1 Interns

Among the interns there are mostly just two people having a conversation, when discussing the user stories. One of the interns (I-1) has very good technical insight into everything that will be done in sprint 2, where the baseline is set. Because of this he leads the discussion for the most part before the team settles on a baseline.

> I-1: This is stories that can be solved using Django?
>
> M: That's right.
>
> I-1: That could have a huge impact for how much time we need...
>
> —
>
> M: All right, just discuss the stories for a bit.
>
> I-3: Are we trying to find the easiest one?
>
> I-1: Let me explain a bit first. What we have to do is to make a model for the part of the database we will work on. The nice part is that when this is done, we can basically with one line of code get an admin interface.
>
> —
>
> I-3: Do we need them in pdf format then?
>
> I-1: There are python libraries that can convert html to pdf - it should not be a problem.

It is probable that the interns ended up with a couple of estimates that are too low because of the high technical knowledge of I-1 make them seem trivial. After the technical discussion of the user stories the baseline is apparently chosen on a whim:

> I-5: If we say that the admin stories are the biggest? [2.1 and 2.2]
>
> I-3: Maybe we can use 8 as a baseline? More to go on in each direction? How about the first admin story? [2.1]

Compared to actual effort, the baseline is not a medium sized user story, as assumed, but a small user story. User story 2.1 and the similar user story 2.2 are overestimated as a result. I-5's comment that user story 2.1 and 2.2 are the biggest might have had an anchoring effect on the others resulting on underestimating the rest of the user stories.

### 6.3.2 Employees

The employees also have some technical talk when they discuss the user stories, E-1 has very good technical knowledge concerning the user stories, yet this is not made the center of the discussion. The employees have estimated in groups before and are more familiar with the setting. They are aware of some of the culprits of group estimation:

> E-1: The three report stories will get the same estimate.
>
> E-2: Don't make any assumptions before we have started.

The baseline found by the employees seems to be the result of identifying a medium sized user story correctly (compared to the actual hours used):

> E-3: Ok, then we have, the admin stories are kind of medium size?
>
> E-2: Could this one be 5? [points at story 2.2], I mean the other one [2.1] has less relations.
>
> E-1, E-3: Sure.

### 6.3.3 Thoughts on baseline discussion

As is outlined in this section it seems important that the baseline user story really has the relative effort assumed compared to the other stories, that the user stories assumed to be bigger and smaller really are. If not, a scenario like the one that happened to the interns is possible.

How the baseline discussion is done, can also have an impact on how the user stories are later estimated. In retrospect it seems like some of the good qualities of planning poker are lessened if the user stories are discussed in too much details, possibly resulting in early anchoring effects. With this in mind, and assuming a group estimation technique like planning poker is used, it is probably a good idea to keep initial discussion of user stories brief.

The complete discussion of user stories before the baseline is set can be found in Appendix C.

# 7 Conclusions and further work

From study and analysis of a four week project using two teams to provide estimates of user stories, the data point in the direction that relative estimates are more consistent. This suggests that using relative a estimation method to estimate user stories will enable more accurate plans to be derived from the estimates than for a similar absolute estimation method.

The data also suggest that using a relative estimation method improved estimation performance when estimating the work of others, compared to an absolute estimation method.

Because the size of the dataset is small, it is important to interpret the results carefully.

Further research should include studies on larger projects, with enough user stories to allow more methodology choices and result in more robust analyses.

To look at a reversal of the situation to what is done in this thesis could be interesting; by comparing estimates of user stories that has about equal amount of actual hours.

# References

[1] Mike Cohn. *Agile Estimating and Planning.* Prentice Hall PTR, 2006.

[2] W. Edwards Deming. *Out of the Crisis.* Massachusetts Institute of Technology, Center for Advanced Educational Services, 1986.

[3] Kent Beck et. al. Manifesto for agile software development. www.agilemanifesto.org, 2001.

[4] James Grenning. Planning poker or how to avoid analysis paralysis while release planning. 2002.

[5] The Standish Group. The chaos report, 1994.

[6] Nils C. Haugen. An empirical study of using planning poker for user story estimation. In *AGILE '06: Proceedings of the conference on AGILE 2006*, pages 23–34, Washington, DC, USA, 2006. IEEE Computer Society.

[7] Nils Christian Haugen. Moderne systemutvikling og estimering. In *Simula estimeringsseminar*, 2007.

[8] Harry Helson. *Adaptation-level theory.* Harper & Row, 1964.

[9] Magne Jørgensen and Kjetil Johan Moløkken-Østvold. How large are software cost overruns? critical comments on the standish group's chaos reports. *Information and Software Technology*, 48(4):297–301, 2006.

[10] D. R. J. Laming. *The Measurement of Sensation.* Oxford University Press, 1997.

[11] Craig Larman and Victor R. Basili. Iterative and incremental development: A brief history. *Computer*, 36(6):47–56, 2003.

[12] Albert L. Lederer and Jayesh Prasad. Nine management guidelines for better cost estimating. *Commun. ACM*, 35(2):51–59, 1992.

[13] George Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information, 1956.

[14] Kjetil Moløkken and Magne Jørgensen. A review of surveys on software effort estimation. In *ISESE '03: Proceedings of the 2003 International Symposium on Empirical Software Engineering*, page 223, Washington, DC, USA, 2003. IEEE Computer Society.

[15] Kjetil Moløkken-østvold and Magne Jørgensen. A comparison of software project overruns-flexible versus sequential development models. *IEEE Transactions on Software Engineering*, 31(9):754–766, 2005.

[16] Ken Schwaber. Scrum development process. *Advanced Development Methods.*

[17] F.W. Taylor. *The Principles of Scientific Management.* Harper and Brothers Publishers, 1911.

[18] E. Yourdon. *Death March: The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects.* Prentice Hall, 1997.

[19] Helen Sharp Yvonne Rogers and Jeniffer Preece. *Interaction Design: beyond human-computer interaction.* Wiley, 2007.

[20] Kjetil Moløkken Østvold et. al. Using planning poker for combining expert estimates in software projects. *The Journal of Systems and Software*, 2008.

# List of Tables

## List of Figures

# Appendices

Appendix A contains the user stories used in the study project.

(page 57)

Appendix B contains the raw data from the planning poker sessions.

(page 59)

Appendix C contains corrections and notifications for collected data and excerpts from various discussions of relevance.

(page 63) ´

# Appendix A

Appendix A contains the user stories translated from Norwegian for all the sprints.

## User stories sprint 1

1.1 As an employee I want an overview of hours reported for each month.

1.2 As an employee I want to log in with Leancast.

1.3 As an employee I want to reset the comments for each week.

1.4 As an employee I want to report hours for a week.

1.5 As an employee I want to write comments for each week.

1.6 As an employee I want to report hours on different projects/codes.

1.7 As an employee I want to see an overview of hours reported for one week.

## User stories sprint 2

2.1 As an administrator I want to add, deactivate and update: Employees, projects, customers and departments.

2.2 As an administrator I want to add, deactivate and update: Employees project assignments, projects customer relations and employees department relation.

2.3 As an administrator I want to generate reports grouped on customers so that I can complete invoicing.

2.4 As an administrator I want to generate reports grouped on employees as a basis for salaries.

2.5 As an administrator I want to generate reports grouped on projects so that I can do budget follow up.

2.6 As an administrator I want to generate reports in CSV-format so that i can import them into a spreadsheet.

## User stories sprint 3

3.1 As an employee I want to see my flex time account.

3.2 As an employee I want to register hours for overtime.

3.3 As an employee I want to mark registered hours for a month as done.

3.4 As an administrator I want to change locked status on hours so that employees can correct mistakes.

## User stories sprint 4

4.1 As an administrator I want to create labels, so that reported hours can be marked in different ways.

4.2 As an employee I wish to mark hours and get an overview of the "balance" of the label, so that I always can see available time for any given label.

4.3 As an administrator I want to see hours reported by all employees with a month and week view.

4.4 As a company we wish to limit the number of self certified sick leave days to be in line with laws and regulations.

# Appendix B

Appendix B contains the raw data from the planning poker session. Each number represents one card, and each row represents one round of planning poker. In some cases consensus was reached even if not everyone played the same card, this was due to agreement after discussion of the lowest and highest card. Two user stories are marked with (Baseline) in the consensus column, this was where the teams choose the baseline in sprint 2.

In sprint 1 with the interns (table 14) some of the numbers might not be accurate due to problems with the video. All consensus number are correct however, as they were recorded separately in addition to the video.

## Sprints using Ideal hours (absolute)

| User story | I -1 | I -2 | I -3 | I -4 | I -5 | Consensus |
|---:|---:|---:|---:|---:|---:|---:|
| 1.1 | 13 | 20 | 13 | 5 | 5 | |
| | 8 | 13 | 8 | 8 | 13 | |
| | 8 | 13 | 8 | 8 | 13 | 8 |
| 1.2 | 1 | 0.5 | 2 | 1 | 5 | |
| | 1 | 0.5 | 1 | 1 | 2 | 1 |
| 1.4 | 20 | 20 | 13 | 20 | 13 | 20 |
| 1.6 | 3 | 13 | 8 | 20 | 5 | |
| | 8 | 5 | 8 | 13 | 8 | 8 |
| 1.7 | 8 | 13 | 8 | 8 | 5 | |
| | 8 | 8 | 8 | 8 | 8 | 8 |

Table 14: Sprint 1 - Interns

| User stroy | I -1 | I -2 | I -3 | I -4 | I -5 | Consensus |
|---|---|---|---|---|---|---|
| 3.1 | 20 | 20 | 8 | 8 | 8 | |
| | 13 | 20 | 13 | 13 | 13 | 13 |
| 3.2 | 5 | 13 | 20 | 13 | 5 | |
| | 13 | 8 | 13 | 13 | 8 | 13 |
| 3.3 | 5 | 5 | 5 | 3 | 3 | |
| | 5 | 5 | 5 | 3 | 3 | 5 |
| 3.4 | 3 | 5 | 3 | 3 | 2 | 5 |

Table 15: Sprint 3 - Interns

| User story | E-1 | E-2 | E-3 | E-4 | E-5 | Consensus |
|---|---|---|---|---|---|---|
| 1.1 | 20 | 8 | 13 | 13 | 20 | |
| | 20 | 20 | 20 | 20 | 20 | 20 |
| 1.2 | 0 | 0 | 1 | 1 | 2 | 0 |
| 1.4 | 13 | 40 | 13 | 20 | 20 | |
| | 13 | 40 | 40 | 20 | 20 | 40 |

Table 16: Sprint 1 - Employees

| User story | E-1 | E-2 | E-3 | E-6 | Consensus |
|---|---|---|---|---|---|
| 3.1 | 40 | 40 | 20 | 40 | 40 |
| 3.2 | 40 | 20 | 20 | 40 | 40 |
| 3.3 | 13 | 13 | 20 | 13 | |
| | 13 | 13 | 13 | 13 | 13 |
| 3.4 | 5 | 8 | 5 | 8 | |
| | 5 | 5 | 5 | 5 | 5 |

Table 17: Sprint 3 - Employees

## Sprints using story points (relative)

| User story | I -1 | I -2 | I -3 | I -4 | I -5 | Consensus |
|---:|---:|---:|---:|---:|---:|---:|
| 2.1 | | | | | | (Baseline) 8 |
| 2.2 | 8 | 13 | 8 | 13 | 13 | 13 |
| 2.3 | 5 | 5 | 5 | 5 | 5 | 5 |
| 2.4 | 3 | 13 | 5 | 5 | 13 | |
| | 5 | 5 | 5 | 5 | 5 | 5 |
| 2.5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 2.6 | 3 | 8 | 8 | 5 | 3 | 3 |

Table 18: Sprint 2 - Interns

| User stroy | I -1 | I -2 | I -3 | I -4 | I -5 | Consensus |
|---:|---:|---:|---:|---:|---:|---:|
| 4.1 | 40 | 13 | 40 | 20 | 40 | |
| | 40 | 40 | 100 | 40 | 40 | 40 |
| 4.2 | 8 | 13 | 13 | 13 | 8 | 13 |

Table 19: Sprint 4 - Interns

| User story | E-1 | E-2 | E-3 | Consensus |
|---:|---:|---:|---:|---:|
| 2.1 | 3 | 3 | 3 | 3 |
| 2.2 | | | | (Baseline) 5 |
| 2.3 | 8 | 13 | 13 | 13 |
| 2.4 | 13 | 13 | 13 | 13 |
| 2.5 | 13 | 13 | 13 | 13 |
| 2.6 | 8 | 3 | 8 | |
| | 2 | 3 | 5 | |
| | 2 | 3 | 3 | 3 |

Table 20: Sprint 2 - Employees

| User story | E-1 | E-2 | E-3 | E-4 | E-5 | Consensus |
|---:|---:|---:|---:|---:|---:|---:|
| 4.1 | 20 | 13 | 40 | 100 | 20 | |
| | 40 | 40 | 40 | 40 | 40 | 40 |
| 4.2 | 40 | 100 | 20 | 20 | 20 | |
| | 40 | 100 | 40 | 40 | 40 | 40 |

Table 21: Sprint 4 - Employees

# Appendix C

Appendix C contains corrections and notifications for collected data and larger excerpts of discussion during the estimation sessions translated from Norwegian.

Text in [square brackets] are clarifications.

Text in regular (parentheses) are actions.

## Corrections and notifications for collected data

### Sprint 1:

The employees found user story nr 1.6 and 1.7 to be implicit under user story 1.4, meaning that the estimate for user story 1.4 would also include the work for user story 1.6 and 1.7. To accommodate for this, the interns estimates and reported hours for user story 1.4, 1.6 and 1.7 are added together and seen as a whole when comparing to the employees estimate of user story 1.4. The employees also took user story 1.3 out of the sprint.

### Sprint 2:

No corrections or notifications for sprint 2.

### Sprint 3:

In sprint 3 a rather large database conversion took place. The time reported on the database work affects the implementation of 3 user stories (3.1, 3.2 and 3.3). The interns did not know which user story to report the hours and simply reported the extra work on the database separately. To accommodate for this, each of these user stories had 1/3 of the database time added to their reported hours to reflect actual hours used.

User story 3.3 caused a lot more trouble to implement than anyone had anticipated; it was not finished in the original sprint, but in sprint 4. Reported hours from sprint 4 on this user story were added to the reported hours from sprint 3 to reflect actual hours used.

**Sprint 4:**

Sprint 4 was mostly used to tie up some loose ends from the previous sprinst and not much new work was done. One new user story (4.2) was completed, however the functionality delivered is not the same as the user story states. After getting one of the employees to give an impartial look at the video from the estimation sessions, he agreed that that what was discussed and estimated corresponds to the implementation. This user story is therefore judged representative for the study and included.

## Employees takes out user story 1.3

*User story 1.3*

E-5: This does not make any sense at all?

E-2: I think it is because Ehour has its own reset button.

E-3: Did any of you ever use that button?

E-1: This is just plain browser functionality.

M: Ok, let's take this one out.

## Employees adding user story 1.6 , 1.7 in their estimate for 1.4

*User story 1.6*

M : (reads the user story)

E-1: Didn't we already do that?

E-5: What is the difference for registering an hour and register hours on different posts?

E-4: Did we misunderstand the last one?? [story 1.4]

(People talking over each other, quickly calms down)

E-3: Then we must say that the last one was view, and this is implementation.

E-5: Let's just strike this one out

E-4: The question is if it should be split in two.

M: The interns interpreted it like this - implement for one project first, then for many.

E-4: That won't make much of a difference.

E-4: Ok, maybe we should drop it.

E-1: I thought we all agreed the functionality was under the last one [story 1.4]

(nods, and grunts of approval)

M: Ok, next time we will go through all the user stories first.

(more nods)

M: All right, then we will take that out.

E-2: Maybe we should do our estimates before the interns?

E-3: Then we can discuss the quality - these user stories were made a little hastily last week.

*User story 1.7*

M: Ok, last user story (reads the story).

E-3: This was also included in the last one [story 1.4]

M: Ok, let's take this one out as well then.

E-2: Did they take all this in the backlog? Cool - I can't wait to register hours in Leancast next week!

M: They might be a little overcommitted.

E-3: We'll leave it to them to figure that out, if it's a problem.

## Discussions of user stories before baseline is set

### Employees

M: Ok, let's try to find a baseline for these stories, in story points. Try to pick the user story that is about in the middle of these stories. Since there are so few stories, I think we should just choose one that is 5, and go from there.

E-3: Is 5 in the middle?

E-2: Is it relative estimating now?

M: Yes this is relative estimating, now we are using story points.

E-3: I'm not used to these cards...

M: There are a few cards in the deck that we probably won't use, like the infinity card. You can use any card, if you feel it's the right one.

E-1: So, we do not assume the order of which work is being done? That the interns don't know things must be spread evenly out over the user stories?

M: Well, yes.

E-3: Now we actually have four tasks that are of equal size...

E-1: The three report stories will get the same estimate.

E-2: Don't make any assumptions before we have started.

M: That's part of the point here, we might not agree right away, then we can discuss and play more rounds.

E-1: Well in reality, the first rapport will take more time and the others will be much quicker to finish. But we will give them the same estimate?

M: Now it's just relative size of the user stories in comparison to each other. We're not as interested in how much time each of these will actually take individually to solve.

E-3 Ok, well I think these (points at user story 2.1. and 2.2) might be bigger... than the admin interface?

[Tasks are sorted in three groups: 1: the admin interface, 2.1 and 2.2. 2: the reporting user stories 2.3, 2.4, and 2.5. 3: The conversion story 2.6.]

M: If you think 5 is the wrong number for a user story for a baseline, feel free to pick another number.

(pause 5 sec)

E-3: ok...

(pause 5 sec)

E-2: should we start with the first one?

M: No, just look for the user story that will be your baseline, try to find a medium size one.

E-2: oh right...

M: Just to have a reference point.

E-1: I'm unsure which one is the hardest...

(pause 10 sec)

E-1: This side or this side? Which is larger (points at the group of admin user stories, and the other group) Here it is like; they must look at what fields there are. They don't need to think so much here, they just need to include all that is in the model, it's not that much work. They still have to do it though, the modeling in Django...

E-2: Ok, since i don't know anything about Django at all, could you explain a bit what work needs to be done here?

E-1: (Technical talk on Django)

E-3: Ok, so what do you need for the reports?

E-1 : You need to make a page with form, now I'm thinking it is a bit more complete stack, it's more of controller and view...

E-3: Ok, then we have, the admin stories are kind of medium size?

E-2: Could this one be 5? [points at story 2.2], I mean the other one [2.1] has less relations.

E-1, E-3: Sure.

## Interns

M: Ok, first I'll read the user stories, then we will try to find a baseline (reads the user stories).

I-1: This is stories that can be solved using Django?

M: Thats right.

I-1: That could have a huge impact for how much time we need...

M: All right, just discuss the stories for a bit.

I-3: Are we trying to find the easiest one?

I-1: Let me explain a bit first. What we have to do is to make a model for the part of the database we will work on. The nice part is that when this is done, we can basically with one line of code get an admin interface. So when the model is done, both admin stories [2.1 and 2.2] will practically be done. I've taught 6-7 other people how to do this before, they had a bit more time but we will work on this every day, it should be no problem.

I-3: How is it with implementation between Django and eHour?

I-1: We can do everything in Phyton...

I-3: How will this be compared to the rest?

I-1: Independent, except it is the same database.

I-3: So it is just a link to the another side that is controlled by Django?

I-1: Yes, basically.

I-3: So this is how we make the default page. How is it to change?

I-1: Quite easily customizable. And for the reports, the same way as you have hibernate, there is an ORM layer in Django so we can ask the domain model queries. We can use that to generate the reports.

I-3: Do we need them in pdf format then?

M: No you just generate the report on screen for now, and that they can be exported to CSV [story 2.6]

I-1: There are python libraries that can convert html to pdf - it should not be a problem.

M: you know what CSV is right?

I-2: Comma separated value...

I-1: Python has its own CSV module that does all the hard work for us. That should make this easy for us.

M: Do you have a feeling for what should be the baseline? Do you want to use 5? Don't think in time, now it's just a matter of finding a medium size user story.

I-1: Do you include training in the estimates? Or can we assume that everyone knows Django...

M: You must base this from your own posistion, but at the same time, you need to agree on a story point value for each user story. Try to just focus on the relative size of each story.

I-1: So its relative estimation on this one. Mhm.

(shuffling of paper and some more technical discussion)

I-5: If we say that the admin stories are the biggest? [2.1 and 2.2]

I-1: Both these stories need the data model.

M: Its part of the challenge to not assume the order of which user stories are worked on, try to think of them separately. Don't analyze too much, try to set a baseline, and we can discuss more as we proceed.

I-3: Maybe we can use 8 as a baseline? More to go on in each direction? How about the first admin story? [2.1]

(The team agrees and moves on to planning poker)