UiO : Department of Informatics
University of Oslo

Does Limit on Work-In-Progress (WIP) matter?

Truls Skeie

Master's Thesis Spring 2014

Does Limit on Work-In-Progress (WIP) matter?

Truls Skeie

11th March 2014

Abstract

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Research Question 2

1.3 Approach 2

1.4 Contribution 3

1.5 Chapter overview 3

2 Background **5**

2.1 Waterfall 5

2.2 Scrum 6

2.3 Lean 7

2.4 Kanban 9

2.4.1 Kanban Board 10

2.4.2 WIP limit 11

2.4.3 Benefits with WIP limit according to various articles 12

2.4.4 Limit WIP vs. Unlimited WIP 13

2.5 Lead time . . . 14
2.6 Just-In-Time . . . 15
2.7 Throughput . . . 16
2.7.1 Example of throughput measurement . . . 16
2.8 Code churn . . . 17
2.9 Software Innovation . . . 17

3 Research Methods **19**
3.1 Case study . . . 19
3.2 Choice of case . . . 20
3.2.1 Software Innovation's development process . . . 21

4 Data collected and calculations **23**
4.1 The columns . . . 24
4.2 WIP per day . . . 26
4.2.1 Step 1: Gather all unique dates into a Arraylist . . . 26
4.2.2 Step 2: Gather the remaining dates . . . 27
4.2.3 Step 3 Measure WIP . . . 28
4.2.4 Example . . . 29
4.3 Bug . . . 32
4.3.1 Add Bug . . . 33
4.4 Throughput . . . 34
4.5 Churn . . . 34

4.6 Lead time . . . 35
4.7 Lead time and churn . . . 35
4.8 Outcome . . . 36
4.9 SPSS . . . 36

5 Results **37**

5.1 Introduction . . . 37
5.2 Team 1 . . . 37
5.2.1 WIP correlation . . . 38
5.2.2 Throughput . . . 43
5.2.3 Bugs . . . 45
5.3 Team 2 . . . 46
5.3.1 WIP . . . 46
5.3.2 Lead time . . . 47
5.3.3 Analyzed data per quarter . . . 48
5.4 360 . . . 53
5.5 Krypton . . . 58
5.6 Neon . . . 62

6 Discussion **67**

7 Conclusion **69**

List of Figures

2.1 Waterfall model 6

2.2 Example of a Kanban board 11

2.3 JIT example 15

4.1 Illustrating the WIP timeline for this example 30

5.1 WIP, Throughput, Lead time and Bugs for 360 per quarter 48

5.2 WIP, Throughput, Lead time and Bugs for Frontend per quarter 49

5.3 WIP, Throughput, Lead time and Bugs for Krypton per quarter 50

5.4 WIP, Throughput, Lead time and Bugs for Neon per quarter 52

List of Tables

2.1 Throughput 16

3.1 Excerpt from the data set 20

4.1 The standard of the data set 24

4.2 Information about the columns from the SI dataset 25

4.3 Relationship between variable and columns from SI 26

4.4 Variables of the WIP objects 26

4.5 Showing Task ID, Date From and Date to 30

4.6 Example of how tasks are labeled 32

4.7 How churn is presented in the excel document 35

4.8 How lead time is recorded in the excel document 35

5.1 Correlation - WIP 37

5.2 Team one - Correlation - WIP 38

5.3 Descriptive statistics for WIP - Team one 38

5.4 Descriptive statistic for throughput - team one 39

5.5 Descriptive statistic for throughput feature - team one 40

5.6 Descriptive statistic for throughput bugs - team one 40

5.7 Number of throughput divided into bugs and feature - Team one 40

5.8 The percent number of bugs finished in the same quarter as reported - Team one 41

5.9 Descriptive Statistic for churn feature - Team one 42

5.10 Descriptive Statistic for churn bugs - Team one 42

5.11 Team one - Correlation - Throughput 43

5.12 Descriptive statistic for throughput - team one 43

5.13 team one - Average churn for feature and bugs - Bugs 44

5.14 Lead time for throughput feature and bugs per quarter 44

5.15 Descriptive statistic for churn - team one 45

5.16 Team two - Correlation - Throughput 45

5.17 360 - Descriptive statistic - Bugs 46

5.18 Team two - Correlation - WIP 46

5.19 Descriptive statistic - WIP - Team two 47

5.20 Team one - Correlation - Throughput 47

5.21 Team one - Correlation - Leadtime_union 47

5.22 Descriptive statistic - Throughput - Team two 47

5.23 360 - Correlation 53

5.24 Descriptive statistic - Lead-time and churn - 360 54

5.25 Frontend Correlation 55

5.26 Descriptive statistic - Lead-time and churn - Frontend 56

5.27 Frontend - Descriptive statistic - Bugs 57
5.28 Frontend - Descriptive statistic - TP feature 57
5.29 Frontend - Descriptive statistic - TP bugs 58
5.30 Krypton - correlation 58
5.31 Descriptive statistic - Lead-time and churn - Krypton 59
5.32 Frontend - Descriptive statistic - Throughput 60
5.33 Krypton - Descriptive statistic WIP 60
5.34 Krypton - Descriptive statistic - Bugs 61
5.35 Krypton - Descriptive statistic - TP feature 61
5.36 Krypton - Descriptive statistic - TP bugs 61
5.37 Neon - Correlation 62
5.38 Descriptive statistic - Lead-time and churn - Neon 63
5.39 Neon - Descriptive statistic - Throughput 64
5.40 Neon - Descriptive statistic - WIP 64
5.41 Neon - Descriptive statistic - Bugs 64
5.42 Neon - Descriptive statistic - TP feature 65
5.43 Neon - Descriptive statistic - TP bugs 65

Preface

Chapter 1

Introduction

In this master thesis the focus will be on limit Work In Progress (WIP) who is one of the principles in Kanban. The focus will be to see what impact the principle has in a development process. In order to do so, a data set gathered by an in house software company in Norway called Software Innovation (SI) is used. SI is a Scandinavian software company that delivers Enterprise Content Management applications.

The data set has already been interpreted by another study. That study investigated Scrum vs. Kanban for SI. For interested reader the case study can be found in the article "Quantifying the Effect of Using Kanban versus Scrum: A Case Study" (Sjøberg, Johnsen and Solberg, 2012).

1.1 Motivation

In software development the processes and methods are important in order to deliver the right product on time. In software development one rarely solve two identical problems for different stakeholders. And the problems are getting bigger and more complex, which means that new processes and methods are introduced and the already existing processes and methods needs to adapt to solve the complex problems in the most efficient ways. The number of popular software development methods (e.g. Extreme programming, Spiral, Scrum and Kanban) emerged in the recent years proves the assumption (Gandomani et al., 2013) (Marko Ikonen et al., 2010).

This is why this thesis will focus on software development methods, because the method in each development project is such a key element. Within software development methods, the main focus of this thesis will be the Kanban method and

the principle to limit Work In Progress (WIP). In Kanban the WIP limit is used to limit the number of tasks each developer can work on at each workflow state to prevent bottlenecks and to ensure flow of tasks through the development cycle (Gandomani et al., 2013) (Marko Ikonen et al., 2010).

There are published various literature on Kanban in software development such as "Kanban: Successful Evolutionary Change for Your Technology Business" (D. J. Anderson, 2010), "Kanban and Scrum - making the most of both" (Kniberg, 2010) and "Lean Software Management: BBC Worldwide Case Study" (Middleton and Joyce, 2012). Although there is various literature, there is no information on how to apply WIP limit, even though most of the experience Kanban enthusiasts agree that WIP limit is an important principle. There is no research backing the statement. There exist no clear rule or research on how to apply WIP limit. The literature states that one should experiment with WIP limits in order to find the best WIP limit for one's case (M. Ikonen et al., 2011) (Kniberg, 2010).

Since there is lack of available research on WIP limit my motivation is to investigate WIP limit in this thesis.

1.2 Research Question

In this thesis the overall research question will be to study the effects of WIP limits for an in house software company, and to see if the research will give answers to the following question:

- Do WIP limit in software development matter?
- How to best find the optimal WIP limit?
- Which parameters to consider in order to optimize WIP?

1.3 Approach

This thesis will use case study as an approach in order to answer the research questions. The case study contains a data set from an in house software company. The data set will be calculated and analyzed. The first part of the analyze will be done by a program. The program was made for this thesis in order to calculate the data set. The program will turn the data set from raw quantitative into more suited quantitative data.

The analyzed data will be evaluated on team level. The most valuable teams will be picked and their data will be extracted form the data set by the program. The teams will be evaluated on the basis of number of task labeled with their name in the data set. After the program has calculated each team, statistical analysis will be used to compute correlation and descriptive statistic.The correlation and descriptive statistic will be represented in chapter 5.

1.4 Contribution

My contribution on the research on WIP and WIP limit consist of creating a Java-based program that will measure WIP for each day based on a data set. For the program to work, the data set given as a parameter to needs to have the same format as the data set in this thesis.

1.5 Chapter overview

Chapter 1: Introduction:
The first chapter will consist of a section where the motivation behind this thesis is stated. A section where the research questions this is stated. A section which gives a brief description of how I will approach the data set in order to answer the research questions and a section where my contribution to the research of WIP limit will be stated.

Chapter 2: Background:
Chapter 2 consist of background information which gives a brief introduction to concepts and methods in software development as well as the in house software company, Software Innovation. These methods and concepts are referred to later in the thesis.

Chapter 3: Research Methods:
Chapter 3 introduces the research methods used in the thesis as well as complementary information about Software Innovation and why the data set from Software Innovation is used in this thesis.

Chapter 4: Data collected and calculations:
Chapter 4 gives information about the data set and calculation. Complementary information about how the program operates is given in the chapter as well as information about how the output data from the program is measured using statistical analysis.

Chapter 5: Results:

Chapter 6: Discussion:

Chapter 7: Conclusion:

Chapter 2

Background

In this chapter there will be a brief introduction to Waterfall (Section 2.1), Scrum (Section 2.2), Lean (Section 2.3) and Kanban (Section 2.4) with affiliated tools. There will also be given a brief introduction to the software development company Software Innovation (Section 2.9)

2.1 Waterfall

"The waterfall model is the classical model of software engineering. This model is one of the oldest models and is widely used in government projects and in many major companies" (Munassar and Govardhan, 2010).

The main goal of the waterfall model is to plan in early stages to ensure design flaws before coding is started. Since planning is so critical in the waterfall method it fits project where quality control is a major concern (Munassar and Govardhan, 2010).

The waterfall method consist of several non-overlapping stages as shown in figure 2.1. The figure shows the waterfall model with a life cycle of establishing system requirements and software requirements and continues with architectural design, detailed design, coding, testing and maintenance (Munassar and Govardhan, 2010).

One of the main principles of the waterfall method discourage return to an earlier phase. For example returning from detailed design to architectural design. However, if returning to an earlier phase is needed, it involves costly rework. When a phase is completed the phase requires formal review and extensive documentation

development. Therefore, if something is missed out the architectural design phase it is expensive to correct later (Munassar and Govardhan, 2010)

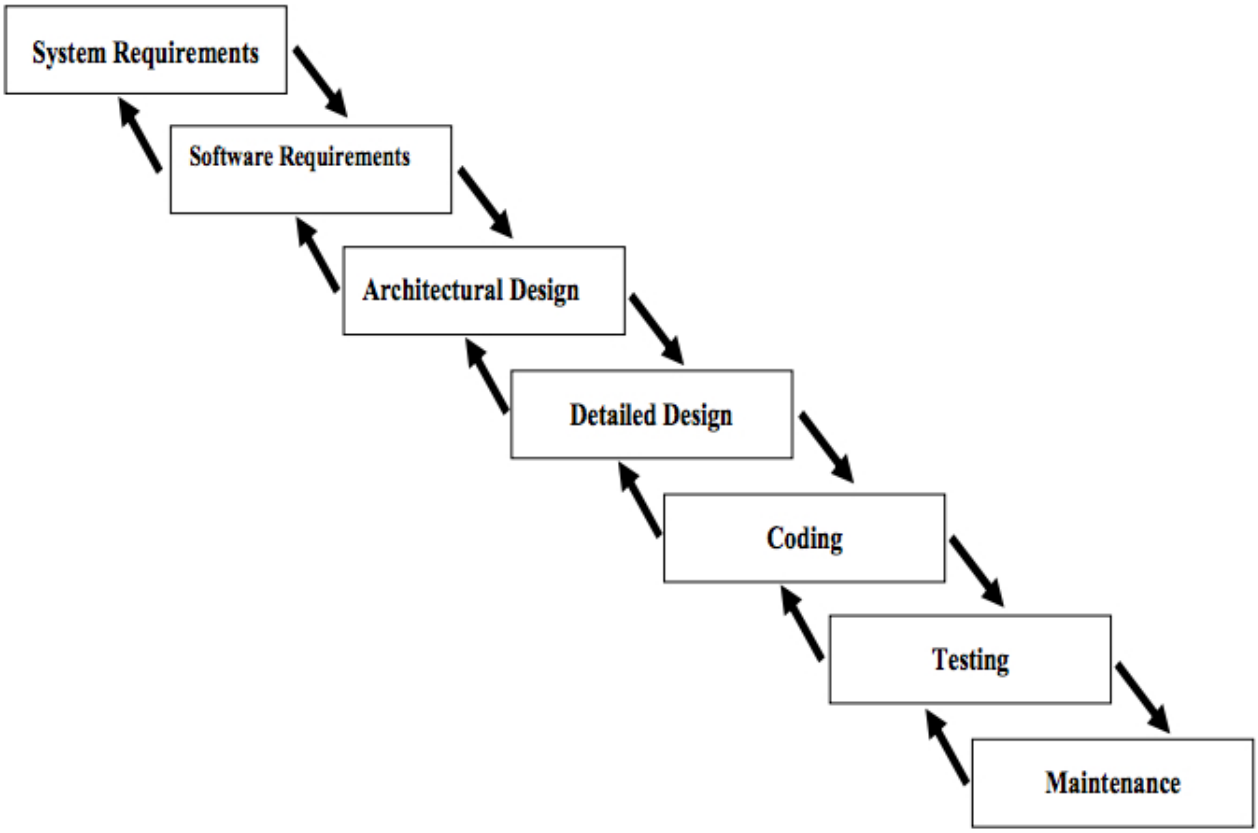


Figure 2.1: Waterfall model

2.2 Scrum

"Scrum is the best-known of the Agile frameworks. It is the source of much of the thinking behind the values and principles of the Agile Manifesto, which form a common basis for all of these approaches" (Alliance, 2012).

The Scrum method relates to value as "Individuals and interactions over processes and tools", "Working software over comprehensive documentation", "Customer collaboration over contract negotiation" and "Responding to change over following a plan" from the Agile Manifesto(Alliance, 2012). These principles are the opposite of the principles of the Waterfall method.

Scrum have three main roles, the Product Owner, the Scrum Master and the members of the development team. The Product owner in collaboration with the Scrum Master decides which work to be prioritized in the backlog. The backlog represents the tasks to be done in order to complete the project. The Scrum Master acts like a team leader and helps the development team and the organization to take best advantages of Scrum. The development team works on tasks specific for the sprint there in (Alliance, 2012).

Sprint is a time-boxed interval over a given time. The Scrum framework suggests duration of sprints to be from one to four weeks. Before each sprint, a sprint planning meeting is conducted with all the team members attending. A Sprint planning meeting is held so the team can discuss tasks from the backlog and come to an agreement of which tasks to be put in the minimal backlog (Alliance, 2012).

In each sprint a minimal backlog is created so the developer knows which tasks to work on in the current sprint. The Product Owner and the team members discuss and decide which tasks from the backlog to be added to the minimal backlog. After the minimal backlog is complete, the Product Owner and the team members discuss each task in order to get a better and shared understanding of what is required in order to complete the tasks. Sprint planning meeting is held so the team can discuss tasks from the backlog and come to an agreement of which tasks to be put in the minimal backlog. (Alliance, 2012).

One of the main principles in Scrum is that it requires that a new feature is ready for release after each sprint. The feature should be a visible part of the product in order to get feedback from end-users. So all the tasks in the minimal backlog combined should be a visible part of the product. (Alliance, 2012).

2.3 Lean

"Lean is all about getting the right things to the right place at the right time the first time while minimizing waste and being open to change" (Raman, 1998)

The Lean approach was introduced between 1948 and 1975 in manufacturing in Japan. Lean strives to maximize the value produced by an organization and delivered to costumer. This is done by finding and eliminate waste, controlling variability and maximizing the flow of delivered software all within the culture of continuous improvements (D. Anderson et al., 2011). In 2003 Mary and Tom Poppendieck first introduced Lean thinking to software development, they published "Principles of Lean Thinking" (M. Poppendieck and T. Poppendieck, 2003). Poppendieck stated that an important tool to manage work flow is the concept of pull-systems, which means tasks are put in production only when a costumer asks for it (M. Poppendieck and

T. Poppendieck, 2009). The pull-system based method Kanban, has in the recent years been introduced more and more to software development, and is becoming one of the keys to Lean practice in software development (D. Anderson et al., 2011).

There are eight fundamental principles in Lean. The eight principles are:

1. **Start Early:** Don't wait for details. As soon as enough information is gathered start the development activity. Get everyone involved in figure out the details. Don't build any walls between people, make people collaborate and start a two-way communication as soon possible. This will start the learning cycle as well.

2. **Learn Constantly:** Start with a breadth-first approach, explore multiple options. The system is expected to change, so focus on creating simplicity code and robustness so the system is easy to change

3. **Delay Commitment:** In order to delay commitment, automated testing and refactoring are essential for keeping code changeable.

4. **Deliver Fast:** Deliver fast mark of excellent operational capability. The whole idea of **delaying commitment** is to make every decision as late as possible when one have the most knowledge.

5. **Eliminate Waste:** The only thing worth doing is deliver value to the costumer, anything else is waste. See waste and eliminate it is the first key of Lean. Lean suggests using a value stream map for removing waste. A Value Stream Map (VSM) is a map over the whole company chain. VSM helps visualize where waste are located within the company.

6. **Empower the team:** When one are going to deliver fast, there is no room for central control. The work environment should be structured so work and workers are self-directing.

7. **Build Integrity In:** Lean software is build with integrity. That's why one of the principles in Lean suggests that test are integrated into software development just as any code, so it becomes a part of the delivered product.

8. **Avoid Sub-optimization:** In software development it's normal to break down a complex problem into small parts of the problem in order to minimize the complexity. If some of the parts are sub-optimized, bottleneck can occur. For example, if ten developers are hired to work on tasks, but only three testers are hired. The development process is sub-optimized since the developers will likely produce more than the tester can test and that will cause bottleneck.

(M. Poppendieck, 2003)

2.4 Kanban

”One can define Kanban software process as a WIP limited pull system visualized by the Kanban board” (D. Anderson et al., 2011).

Toyota production system introduced Kanban as a scheduling system for Lean and just-in-time (JIT) production during late 1940’s and in the early 1950’s in order to catch up with the American car industry. The Kanban method combined with the Lean approach was a success for Toyota. The success was noticed by the software development industry among others. In the last ten years, more and more software development companies have started to implement development methods such as Scrum and Kanban (Conboy, 2009), (Ohno, 2001).

In the recent years, more software projects adapt to Kanban and Lean (D. Anderson et al., 2011), and this is one of the reasons why this thesis will focuses on Kanban and one of it’s main principle WIP limit. One of the most important people in Kanban software development, David Anderson also referred to as ”father of Kanban in the software development industry” (Gupta, 2013) and author of the book "Kanban: Successful Evolutionary Change for Your Technology Business" has stated ”If you think that there was Capability Maturity Model Integration, there was Rational Unified Process, there was Extreme Programming and there was Scrum, Kanban is the next thing in that succession.” (Leonardo Campos, 2013)

The Kanban method splits the problem into many small pieces of problems. When the small pieces are defined by the team, the problems are put up on the Kanban board to visualize the problems, track what others are working on and see potential bottlenecks during development. When people start to understand Kanban, they easily discovered where the bottlenecks are (Shinkle, 2009).

Kanban system focus on:

- Continuous flow of work.
- No fixed iterations or sprints.
- Work is delivered when it’s done.
- Teams only work on few tasks at the time specified by the WIP limit.
- Make constant flow of released tasks.

(D. Anderson et al., 2011).

One of the main differences between Scrum and Kanban is sprint and estimation. Contrary to Scrum, Kanban do not use the principles of sprints. In Kanban the tasks do not need to be estimated or finished within a certain time. In the article "simulation of software maintenance process, with and without a work-in-process limit" (Concas et al., 2013) they did found out that if they let the developers work with small tasks at time and not interrupted, they will be more effective. The authors did found that Scrum was to rigid for the development team. The estimation and sprint meetings worked counterproductive. The authors made the developers change to Lean-Kanban. The change implied the removal of sprints. After removing sprints the teams increased the ability to perform work, lower the lead time and meet the production dates (Concas et al., 2013).

In the article "Quantifying the Effect of Using Kanban versus Scrum:" the company also felt that the Scrum approach was too rigid, and the article also reported positive results when the team changed to Kanban. The company almost halved its lead time, reduced the number of weighted bugs by 10 percent, and improved productivity (Sjøberg, Johnsen and Solberg, 2012). Other article also states that Scrum is too rigid and that's Kanbans advantages over Scrum (Beedle et al., 1999) (Brekkan and Mathisen, 2010) .

2.4.1 Kanban Board

"The Kanban board makes it clear to all the team members the exact status of progress, blockages, bottlenecks and they also signal possible future issues to prepare for"(Middleton and Joyce, 2012).

The Kanban board is one of many tools in Kanban. It's used to control WIP, increase the information flow with visualization and spot bottlenecks (Concas et al., 2013). A Kanban board is illustrated in figure 2.2. Each column has an intuitive name in order to describe itself so the developers easily can track where each task is. In figure 2.2 each column is named "Backlog", "In progress" and "Done". Each column can have a WIP limit to specify how many works in progress there are allowed in the column (Middleton and Joyce, 2012). In figure 2.2 the WIP limit is stated under the column name. The backlog column have a WIP limit of 4, In progress has 5 and done doesn't need a WIP limit. The yellow stickers represents the tasks. Some follow the path to mark stickers with different colors representing the severities. In "Kanban Implementation in a Telecom Product Maintenance" the stickers has three different colors, green, yellow and red depending on how close to overdue the tasks are. If the sticker is red, the task is already overdue, if the tasks are soon-to-overdue its marked with yellow stickers (Seikola, Loisa and Jagos, 2011).

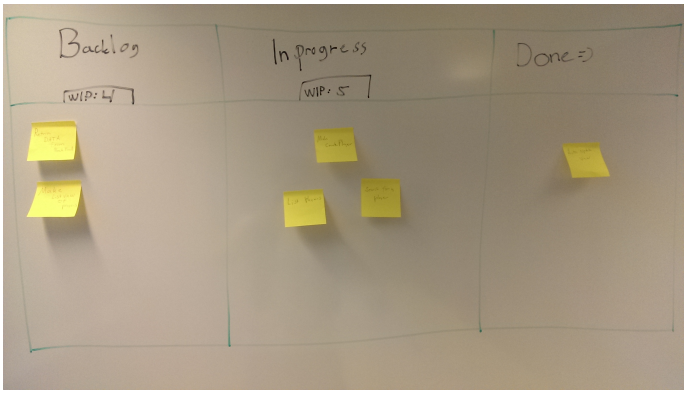


Figure 2.2: Example of a Kanban board

2.4.2 WIP limit

"WIP limits seem to be the worst understood part of the Kanban system. When used properly, it exposes bottlenecks and reduces lead time for individual work items. Used improperly, it can starve developers for work or result in too many people working on the same work items." (Shinkle, 2009)

WIP limit is one of the core principles in Kanban (Seikola, Loisa and Jagos, 2011). WIP limit helps to reduce overhead by limit task-switching for each developer, visualize bottlenecks and make constant flow of tasks throughout the development (D. Anderson et al., 2011). One way to explain WIP and the impact of WIP limit is to use cars and roads as analogy. All roads have it maximum capacity of cars. When this limit is reached, traffic jam occurs and the throughput of cars decreases and lead time increases. The same can be said about software development teams, a software team has a maximum number of tasks they can perform, if the team is pushed over the maximum limit, the throughput of tasks decreases and lead time increases.

When first implementing Kanban, Shinkle explains that the users don't care about WIP or setting a WIP limit, but rather the visibility of Kanban through the Kanban board. When user get more experience with Kanban, they start to attempt the principles of WIP limit (Shinkle, 2009). Srinivasan, Ebbing and Swearing said that setting the WIP limit is not easy. They suggest that the WIP limit is set, and then observe throughput, and adjust after that (Srinivasan, Ebbing and Swearingen, 2003).

The principle of Kanban also says you should limit WIP, but what should the limit be? The principles of Kanban tells us to experiment (Kniberg, 2010). The Lean Software Management article(Kniberg, 2010).and the Impact of Kanban on Software Project Work (M. Ikonen et al., 2011) both suggest that WIP should be minimized as well. The conclusion of present study is to keep the WIP limit low and experiment by slowly

increase the WIP limit until the throughput decreased and lead time increased, then you know that the previous WIP limit was the perfect one.

The importance of limit WIP has been stated by research. Both the articles by Giulio Concas, Hongyu Zhang (Concas et al., 2013) and David Anderson, Giulio Concas, Maria Ilaria Lunesu, and Michele Marchesi (D. Anderson et al., 2011) did a research on the difference between limit WIP and unlimited WIP. A summary of this two articles is shown in section 2.4.4

On how to determine WIP limit one research was found, if one implement Kanban with sprints or uses Scrum, Łukasz proposes to use the effectiveness metric to help determine the WIP limit. The effectiveness metric should be applied after end sprint according to Łukasz. After each sprint, one can apply the effectiveness and the result could be used as a guideline for WIP limit for the next sprint. The effectiveness metric takes the number of bugs found (ai) and the number of bugs found by external people (e.g. lawyers, accountants, coaches, consultants, translators, internal and external service providers etc.) (ei), and minus ai and ei, then divide the result by ai and multiply it by 100% as shown in 2.1 (Sienkiewicz, 2012)

$$Ei = \frac{ai - ei}{ai} * 100\% \tag{2.1}$$

2.4.3 Benefits with WIP limit according to various articles

1. When lowering the WIP limit it will help people avoid task switching. When one is task swtiching it's hard to be able to fully concentrate. (M. Ikonen et al., 2011).

2. There's stated when using short-cycle times and Kanban board to limit WIP, the software development team's learning is increased' (Middleton and Joyce, 2012)

3. In the article "Lean Software Management: BBC Worldwide Case Study" (Middleton and Joyce, 2012) the team started to determine WIP by their constraints and quickly find their bottlenecks; this gave the team more experience in dealing with WIP and increased productivity(Middleton and Joyce, 2012).

4. It helps team to reduce overhead (Mark L. Spearman and J.Hopp, 1990).

5. Decrease lead time (Mark L. Spearman and J.Hopp, 1990).

6. Increase throughput (Mark L. Spearman and J.Hopp, 1990).

7. It reduces flow times (Mark L. Spearman and J.Hopp, 1990).

8. Reduces variation (Mark L. Spearman and J.Hopp, 1990).

9. Improves quality (Mark L. Spearman and J.Hopp, 1990).

2.4.4 Limit WIP vs. Unlimited WIP

In the article by Giulio Concas and Hongyu Zhang one of the result was visible at the end of a simulation, the average of closed tasks was 4145 when the WIP was limited and 3853 when the limit was not limited (about 7% less). The paper concludes their finds; developers are more focused on fixing few issues, because the number of issues they can work on is limited. The developers are more likely to continue on the issue from the day before, rather than starting on another issue, this reduces overhead. When developers start on a new issue, they need to use time to familiarize themselves with the code and the issue. That could create unnecessary overhead if some developer already has done it, but that developer is now working on another issue. The study also showed that limit WIP can improve throughput and work efficiency.(Concas et al., 2013).

The case by David Anderson et al. (D. Anderson et al., 2011) did a simulation of the impact of WIP limit vs. no WIP limit on developers with skills in different activities. The four skill activities from the article where:

1. Design
2. Development
3. Testing
4. Deployment

The article did four different simulations.

1. A simulation with WIP limits and seven developers with skill in two of the four activities.
2. A simulation with no WIP limit and seven developers with skilled in two of the four activities.
3. A simulation with WIP limits and seven developers with skill in all of the activities.
4. A simulation with no WIP limits and seven developers with skill in two of the four activities.

The research paper concluded that the two latter is unlikely in the real world, since there is seldom a whole team with developers skilled in all activities.

When the developers had skill in two out of four activities the WIP limit simulation used 100 days, but the non WIP limit simulation used 120 days. The simulation with WIP limit shown an almost constant flow of features that where completed. While in the same simulation with no WIP limit, the flow of feature was much more irregular. (D. Anderson et al., 2011)

Both the studies on WIP limit vs. No limit and the articles and research shows the importance of WIP limit. If Łukasz's effectiveness metric is disregard, there is no clear rule on how to determine WIP limit even though WIP is a crucial principle in order to maxims the potential of Kanban.

2.5 Lead time

"Lead time is the total elapsed time from when a customer requests software to when the finished software is released to the customer" (Middleton and Joyce, 2012).

Lead time is an essential ingredient when you look for the optimal WIP. Often in project, lead time is split into pieces, so every task has its own lead time. This gives the development teams the advantages to experiment with different WIP's in order to see the different lead times and then measure which WIP that suits this project the best.

The citation by Middleton and Joyce above is close to definition of what lead time is. This definition could be useful for consultancy companies, but for in-house development company with few releases each year this definition is unsuitable. The article "Quantifying the effect of Using Kanban versus Scrum" (Sjøberg, Johnsen and Solberg, 2012) stated two reason why this is unsuitable for in house development companies:

"First, the amount of time a work item remains in the backlog queue before it's put on the board is a function of priority, not whether the company uses Scrum, Kanban, or other development methods. Furthermore, companies that develop and sell products to many customers might propose new features themselves and put them on the backlog before any customers request them. Second, given a policy of two or three releases a year, the result of a work item isn't delivered to the customer immediately after it's finished" (Sjøberg, Johnsen and Solberg, 2012).

2.6 Just-In-Time

"Just-In-Time is based on delivering only the necessary products, to the necessary time and the necessary quantity." (Lai, Lee and Ip, 2003).

Just-In-Time (JIT) was introduced 30 years ago by Toyota in combination with Lean. JIT has been developed to increase productivity through waste reduction and increasing the value added on the production processes. In one of the books by Mary and Tom Poppendieck the JIT principle is explained (M. Poppendieck and T. Poppendieck, 2006). To explain, illustrate and visualize the JIT principle Mary and Tom Poppendieck uses the picture 2.3 (Lai, Lee and Ip, 2003) (M. Poppendieck and T. Poppendieck, 2006).

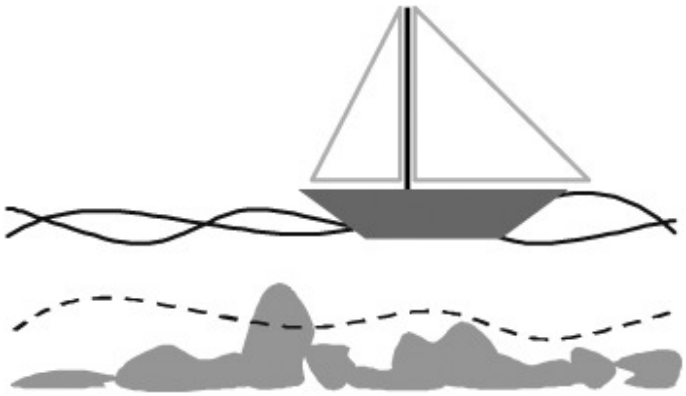


Figure 2.3: JIT example

In picture 2.3 the stream reflects the inventory. Under the stream, there are rocks located in different sizes. The rocks illustrates waste and problems that can occur. If the stream level is lowered, the rocks are more visualized. At this point you have to clear out rocks in order to make the boat continue it's journey, or it will crash into the rocks. After the rocks are cleaned out, one can lower the stream level again and continue until it's just pebbles left

If one lower the stream (inventory), problem and waste will become visible. But why do Lean want to lower inventory in order to make problems and waste occur? Because when problem and waste occurs, you are able to fix the problem and remove the waste. Fixing the problem and removing the waste can have several benefits such as, your process could be optimized and you are on step closer to have zero problems and zero waste. (Lai, Lee and Ip, 2003) (M. Poppendieck and T. Poppendieck, 2006).

2.7 Throughput

”The output of a production process (machine, workstation, line plant) per unit time (e.g., parts per hour) is defined as the systems throughput or sometimes throughput rate” (Adams and Smoak, 1990)

The main concept of throughput is to measure how productive teams, people or companies are. Throughput is measured in number of finished delivered tasks or units per hour, day, week, month, quarter or year. This applies to both software development and manufacturing. But there are some difference. In software development each task is more abstract than in manufacturing and in software development each tasks can have different solutions depending on how the team or developer approaches the task (Rouse, 2005).

A key factor in successfully measuring throughput in software development is to specify the size of each task. If the size is not specified a developer x can have throughput of 1 per week, but another developer could have throughput of 3 and they still have done the same amount of work and this will give wrong results if the throughput is measured. An example of throughput is listed in subsection 2.7.1. (Rouse, 2005)

2.7.1 Example of throughput measurement

This is a simple example to illustrate throughput with different task sizes. Lets say Team x had a throughput of eighteen tasks after the first quarter, twenty after the second, fifteen after the third and twelve after the last quarter and they used Scrum the first two quarters and Kanban the last two as illustrated in table 2.1. It will look like team x benefits most from Scrum. But if the task during the Kanban time was twice the size of Scrum, Kanban would suite team x the best. In order to get valid result from throughput measurement the size of tasks has to be agreed upon by the teams or company.

Quarter	Throughput	Method
1	18	Scrum
2	20	Scrum
3	15	Kanban
4	12	Kanban

Table 2.1: Throughput

2.8 Code churn

"Churn is defined as the sum of the number of lines added, deleted, and modified in the source code" (Sjøberg, Johnsen and Solberg, 2012).

Churn is a measure that's not so quite known as lead time, throughput or WIP. Churn is a term that's used as surrogates for effort in software engineering. Many studies in software engineering use code churn or revisions as surrogate measure of effort (Yamashita, Anda, Mockus et al., 2012). Emam stated that "analysts should be discouraged from using surrogate measures, such as code churn, unless there is evidence that they are indeed good surrogates (El-Emam, 2000)." The authors of "Quantifying the effect of code smells on maintenance effort" agree that one should cautious when using surrogates for effort.

2.9 Software Innovation

Software Innovation is a Scandinavian software company. SI develops and delivers market-leading Enterprise Content Management applications that helps organizations improve and increase efficiency in document management, case handling and technical document control. SI builds products around Microsoft Sharepoint platform and tightly integrated into the Microsoft Office environment (Sjøberg, Johnsen and Solberg, 2012). (*Software Innovation* 2013).

SI has approximately 300 employees in Oslo, Copenhagen and Stockholm and Bangalore (*Software Innovation* 2013). From 2001 to 2006 SI used Waterfall process. In 2007 SI changed to Scrum and in 2010 SI went from Scrum to Kanban.

Chapter 3

Research Methods

In this chapter the research methods will be introduced and the reason why the data from Software Innovation is used in this master thesis will be stated. The first section in this chapter, Section 3.1 will give a brief introduction to the research method "Case Study". Section 3.2 is about the choice of case and complementary information about SI.

3.1 Case study

To answer the research question, a case study will be conducted. A case study is used to explore causation in order to find underlying principles of a study (Shepard and Greene, 2002)(Yin, 2008). But which methods one can use in a case study or how the case study is conducted is ambiguous. It might be that the method is qualitative or quantitate. Or that it utilizes a particular type of evidence (for example ethnographic, participant observation or field research). Jennifer Platt stated: "Much case study theorizing has been conceptually confused because too many different themes have been packed into the idea "case study" "(Gerring, 2006).

John Gerring stated: "A case study may be understood as the intensive study of a single case where the purpose of that study is - at least in part to shed light on a larger class of cases (a population)" (Gerring, 2006). John also stated: "Case connotes a spatially delimited phenomenon (a unit) observed at a single point in time or over a period of time"(Gerring, 2006).

As one can see there is no clear rule what exactly a case study is. In this thesis the case study is used to explore WIP limit's effect in software development. And the purpose

is to shed light on WIP limit in software development and if it matter. In order to do so, the data set from SI with quantitative data analyzed, with data over a period of time (2010-2013).

3.2 Choice of case

The case study in this thesis is based on data set from an in-house software company named Software Innovation. The data set contains information about each task SI has worked on in the last five years (2008-2013). The data set is represented in a excel document, a excerpt of some of the columns in the document is shown in table 3.1.

ID	Type	Created Date	From Day	Date To	Lead Time	Team
3027	Bug	2008-10-07	2008-10-09	2008-10-16	20	Team one
3028	Bug	2008-10-07	2008-10-07	2008-10-08	10	Team six
3029	Feature	2008-10-07	2008-12-30	2008-12-30	105	Team two
3030	Feature	2008-10-07	2008-10-07	2008-10-07	1	Team three
3035	Bug	2008-10-08	2008-11-20	2008-11-28	17	Team five
3037	Feature	2008-10-08	2008-10-19	2008-10-19	7	Team three
3040	Bug	2008-10-10	2008-11-19	2008-11-19	48	Team one

Table 3.1: Excerpt from the data set

In order to try answering the research questions stated in chapter 1.2, the data from SI will be interpreted. The data set contains thirty columns with different data for each task, mostly of this columns are irrelevant for this study but the important columns is stated in table 4.2. The data from SI will be interpreted on team level. SI has about 25 teams. From the data set there are six teams who have worked on the most tasks. The data from these six teams will be used. The data from SI will be analyzed using the program, which will compute the variables shown in table 4.3 for per day for each of the six team. This data will be further analyzed using SPSS. With SPSS the data produced by the program will be interpreted. The interpreted data will be presented in chapter 5.

The reason SI and the data set from SI is analyzed in this thesis is based on the article "Quantifying the Effect of Using Kanban versus Scrum" by Dag Sjøberg, Anders Johnsen and Jørgen Solberg (Sjøberg, Johnsen and Solberg, 2012). In the article the three authors investigated how SI benefitted from Kanban vs. Scrum. Since Dag is the supervisor of this thesis and he had access to the data set, to use the set from SI is a choice of convenience.

3.2.1 Software Innovation's development process

From 2001 to 2006 SI used the Waterfall process with a life cycle of:

1. Design
2. Implementation
3. Testing
4. Deployment for each new release

(Sjøberg, Johnsen and Solberg, 2012).

In 2007 SI examined their development process, which resulted in decision to change to Scrum. Scrum was implemented with the standard elements of Scrum:

- Cross functional teams
- Sprint planning meetings
- Estimation of work items using planning poker
- Daily standup meetings
- Sprints

(Sjøberg, Johnsen and Solberg, 2012).

SI implemented three weeks sprint, after each sprint a fully tested shippable code was ready. In 2010, SI went from Scrum to Kanban. SI felt that Scrum was too rigid and

didn't fit their purpose, they also feared that inaccurate estimation and time boxing gave them longer lead time. SI also saw Scrum planning meetings as waste which reduced productivity and quality (Sjøberg, Johnsen and Solberg, 2012).

SI decided to implemented Kanban in the following manner. When a work item is pulled from the backlog, SI tries to make the item flow through all the stages until it's ready for release. This procedure happens as quick as possible. In order for an item to be ready for release it has to be at a satisfactory quality level, which is defined by SI. SI also implemented WIP limits, if the WIP limit is reached, no new tasks are started until another task is finished which is based on the principle of just-in-time. (Sjøberg, Johnsen and Solberg, 2012).

Chapter 4

Data collected and calculations

This chapter will introduce complementary information about the data set and the program. The first section will contains information about the content of the data set (Section 4.1). Section 4.2 will introduce the algorithm of how the program measure WIP for each day. The subsection 4.2.4 will provide a comprehensive example of how the program measure WIP per day. The consecutively sections reveals the algorithm of how the program measure bugs (Section 4.3) throughput (Section 4.4), churn (Section 4.5) and lead time (Section 4.6) per day. In the last section the statistical analyze program SPSS is introduced (Section 4.9)

For this thesis a program was made in order to measure the variable (shown in table 4.3) per day. The explanation of how the program works will be split into sections so it will be easier to get a understanding of how the program operates. The explanation of how the program works will first contain a detailed description in words, then a description using Pseudocode (JD, 2013) will be provided. The Pseudocode for each section is simplified with only the key elements of the program code.

The following two tables (table 4.3 and table 4.1) shows the five variables that will measured by the program and the latter table shows how quarters, dates and days are represented.

- The Date standard is specified as YYYY-MM-DD.
- All seven days in the week are taken into account when measuring included Saturdays and Sundays
- Quarter of a year is defined as:
  - January, February and March (Q1)
  - April, May and June (Q2)
  - July, August and September.(Q3)
  - October. November and December (Q4)

(Investopedia, 2013)

Table 4.1: The standard of the data set

4.1 The columns

In this section information about the columns in the data set is given. Table 4.2 shows the columns that the program use from the data set. The table contains the name of the columns and a brief description about what the column mean. The two columns in the table are "Columns from data set" and "Description". The "Columns from data set" contains the name of each column used by the program. The "Description" column gives a brief description of each column in the data set.

Columns from data set	Description
Created Date	When a task is put in backlog
Date From	When a given task is pulled out from the backlog
Date to	When a task is finished and ready for release.
Lead Time	The amount of days elapsed from the date the task was created until the tasks has finished
Type	The type column is labeled as either bug or feature depending on the type of the task
Lines added	Number of lines added to a feature or bug
Lines modified	Number of lines modified when working on a feature or bug
Lines deleted	Number of lines deleted from a bug or feature
Team	States the team who has been working on the task.

Table 4.2: Information about the columns from the SI dataset

Table 4.3 contains three columns, "variable", "description" and "columns from SI". The "variable" column displays the five variables that will measured by the program. The "description" column gives a recap of what the variable is and the "columns from SI" column shows which columns from the data set is needed to compute the variable.

Variable	Description	Columns from SI
WIP	The number of items in progress on the given day	Date From and Date To.
Throughput	Number of tasks finished on a given day	Date To
Churn	Lines added, lines modified and lines deleted added together	Lines Added, Lines Modified, Lines Deleted and Date To
Bugs	The number of tasks labeled as Bug and not feature	Type and Created Date
Lead time	The time used on a task, measured in days	Lead time and Date To

Table 4.3: Relationship between variable and columns from SI

4.2 WIP per day

4.2.1 Step 1: Gather all unique dates into a Arraylist

First step of WIP measurement is to create a WIP object with the attributes in table 4.4. The values that are assigned to the attributes are gathered from the data set file. The program creates a WIP object and assigned values to it a shown in listing 4.1 . After the values are assigned the program puts it into the right Arraylist based on the team variable as shown in listing 4.2

Type	Variable name
Date	start
Date	end
String	team
String	processType
int	WIP

Table 4.4: Variables of the WIP objects

1

[1]Arraylist is a resizable array implementation of a list. The Arraylist class provides function for manipulate the size of the array, check the size of the list and convert the list to an array (Oracle, 2013)

Pseudocode step 1:

```
While inputFile != EOF // EOF = End Of file
    WIP = New WIP()
    WIP.start = inputFile.start
    WIP.end = inputFile.end
    WIP.team = inputFile.team
    WIP.processType = inputFile.processType
    WIP.WIP = 1
    FindTeam(WIP)

```

Listing 4.1: Gather all unique dates into Arraylist

Pseudocode step 1:

```
void FindTeam (WIP w)
    if w.team EQUALS "TeamOne"
      TeamOne.add(w)
    if w.team EQUALS "TeamTwo"
      TeamTwo.add(w)
    if w.team EQUALS "TeamThree"
      TeamThree.add(w)
    if w.team EQUALS "TeamFour"
      TeamFour.add(w)
    if w.team EQUALS "TeamFive"
      TeamFive.add(w)
    if w.team EQUALS "TeamSix"
      TeamSix.add(w)

```

Listing 4.2: Gather WIP object to the right data structure

4.2.2 Step 2: Gather the remaining dates

The data set from SI contains dates from 2008 to 2013, but there are some dates missing which table 3.1 shows. For example the date 2008-10-09 is missing. In order to generate WIP for each day the program has to create the dates that's not in the set.

In order to create the remaining dates, the program takes the first date and the last date from each of the teams's Arraylist created in previous section (4.2.1) as shown in listing 4.3 in line one and two. Then the program checks if all the dates between the first date and the last date are in the team's Arraylist. Each of the Arraylist are sorted on date. If the dates are not in the Arraylist, the program will put the date into the Arraylist as show in method on line ten to thirteen. In order to keep the pseudocode simple, the generateWIP method stated in line twelve was omit. The method creates a new WIP object and returns it.

```
1  WIP first = Arraylist.get(0)//points to the first WIP object in the Arraylist
2  WIP last = Arraylist.get(Arraylist.size()-1)//points to the last WIP object
       in the Arraylist
3  Next_date //points to the next date
4  Next_date = first.getDate() // Next_date assigned before iteration
5  while Next_date NOT EQUALS last.getDate()
6    New_date = Next_Date + 1 //Compute the next date
7    AddToArraylist(New_date, first.getTeam())
8    Next_date = New_date
9
10 addToArraylist(Date d, String team)
11   if d NOT CONTAINS IN Arraylist
12     WIP = generateWIP(d, team)
13     Arraylist.add(WIP)
14
```

Listing 4.3: Gather the remaining dates.

4.2.3 Step 3 Measure WIP

The Arraylist from section 4.2.1 and 4.2.2 now contain a WIP object for each date from 2008 to 2013 for each team. In this step the program will loop through the Arraylist, during the iteration each WIP object is extracted from the Arraylist and the WIP is measured. The two methods stated in line ten and eighteen respectively gather the current WIP (method in line ten) and finds out how many finished tasks (method in line eighteen) and returns the result. The result is used in line five to compute the current WIP.

```
lastWIP =  0
for WIP Object IN Arraylist
  if(DateNotMeasured(WIP.getStartDate()) == true)
    WIP_for_this_date = get_current_WIP(WIP.getStartDate())
    WIP_measured = WIP_for_this_date - Nr_of_finishedDates(WIP.getStartDate)
    + lastWIP
    WIP.setWIP(WIP_measured)
    lastWIP = WIP_measured

int get_current_WIP(Date date)
  current_WIP = 0
  for WIP in  Arraylist
    if date EQUALS WIP.getStartDate()
      Nr_of_dates_to_decrement++
return current_WIP

int Nr_of_finished_dates(Date date)
  Nr_of_dates_to_decrement = 0
  for WIP in Arraylist
    if date AFTER WIP.getEndDate() DO
      if date not picked
        Nr_of_dates_to_decrement++
        dateIsPicked(WIP)
return Nr_of_dates_to_decrement

```

Listing 4.4: WIP measurement

4.2.4 Example

This section will provide a comprehensive example of how the programs' WIP algorithm works. Figure 4.1 shows tasks id in the y-axis and dates in the x-axis. The green line indicates the duration of the task. The figure visualize how many WIPs there are for a given date. For example on the date 2008-10-12, tasks 3, 5 and 6 are in progress, which means the WIP is 3 for 2008-10-12.

In this example dates from figure 4.1 and from table 4.5 will be used to illustrate how the algorithm measure WIP

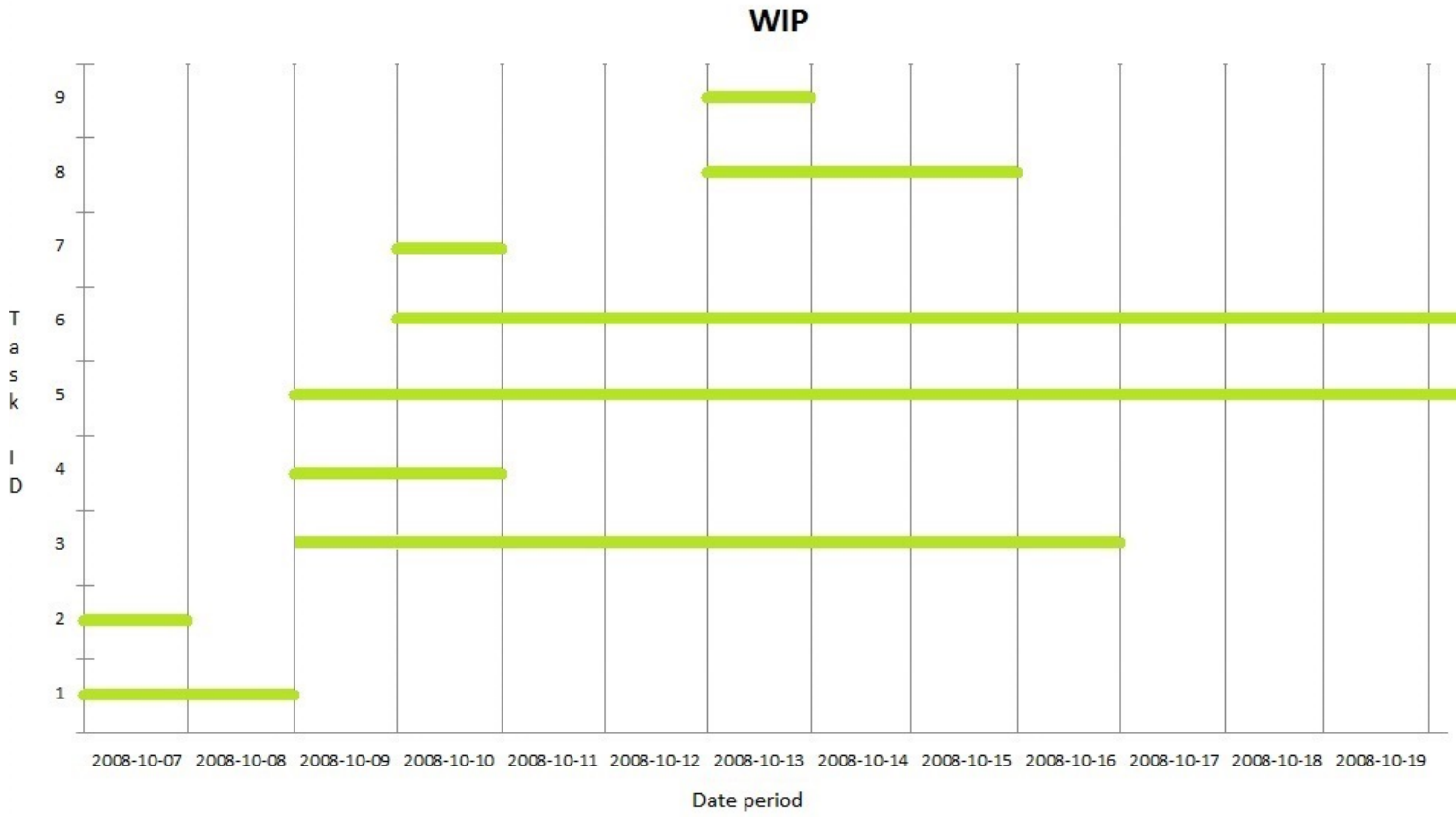


Figure 4.1: Illustrating the WIP timeline for this example

Task ID	Date From	Date To	Team	Process Type
1	2008-10-07	2008-10-08	Team One	Waterfall
2	2008-10-07	2008-10-07	Team One	Waterfall
3	2008-10-09	2008-10-16	Team One	Waterfall
4	2008-10-09	2008-10-10	Team One	Waterfall
5	2008-10-09	2008-11-04	Team One	Waterfall
6	2008-10-10	2008-11-05	Team One	Waterfall
7	2008-10-10	2008-10-10	Team One	Waterfall
8	2008-10-13	2008-10-15	Team One	Waterfall
9	2008-10-13	2008-10-13	Team One	Waterfall

Table 4.5: Showing Task ID, Date From and Date to

4.2.4.1 Step 1

The program will first read in the first line of table 4.5. The first line is the one labeled with task id one. The program creates the WIP-object for line one and it will look like the listing 4.5. The program will follow the exact same procedure until all the dates are read in.

```
1 WIP = new WIP()
2 WIP.start = 2008-10-07
3 WIP.end = 2008-10-08
4 WIP.team = "Team One"
5 WIP.processType = "Waterfall"
6 WIP.WIP = 1
```

Listing 4.5: Creating WIP-object

4.2.4.2 Step two

Now that the whole set has been read in and saved. The next thing to do is to create the remaining dates. The Arraylist contains dates from table 4.5. The program will now extract the first and the last date from the Arraylist. The first date is 2008-10-07 and the last date is 2008-10-13. The program will check if the date after 2008-10-07 contains in the set, which it don't. The program then generates a WIP object for the date 2008-10-08 and adds it to the Arraylist as shown in listing 4.6.

```
1 void createNewWIP(Date d, String team)
2   WIP.start = d
3   WIP.end = d
4   WIP.team = team
5   WIP.processType = "Uknown"
6   WIP.WIP = 0
```

Listing 4.6: Creating WIP-object

4.2.4.3 Step three

The Arraylist now contains the dates from 2008-10-08 to 2008-10-13. The next and last step is to measure WIP for each date. The program will now loop through the Arraylist. The first date is 2008-10-07. The get_current_wip method from line nine in listing 4.4 will be called with the date 2008-10-07 as parameter. The method will return two, since both task one and two where started at 2008-10-07 as shown by figure 4.1. The next to do is to find out how many tasks to decrement the current WIP with. The

method Nr_of_fininshed_dates in line seventeen is called with the date 2008-10-07. As shown by the table 4.5 and figure 4.1 there was no task finished at the date 2008-10-07. So the program updates the WIP objects' counter to two and saves the WIP in the lastWIP object. The next date is 2008-10-08, who the program made in subsection 4.2.4.2. There is no task started at 2008-10-08, but task one is finished at the date. So the Nr_of_fininshed_dates returns one and flag the current date at line twenty-three. The result of WIP_measure in line five is 1 ($0 - 1 + 2 = 1$). WIP at date 2008-10-08 is one, as shown by figure 4.1. The program will continue this procedure until all the dates are measured. The reason why the date is flagged is to be sure that each day only evaluates ones. The if statement listed in listing 4.4 at line twenty-one assurance that.

4.3 Bug

Each task in the data set is label as either feature or bug as shown in figure 4.6. When the program reads in the data file, each task label as bug are saved in a data structure. The code is shown in listing 4.7.

Task ID	Type
57970	Bug
57971	Bug
57972	Bug
57973	Bug
57974	Feature
57975	Feature
57976	Feature
57977	Feature
57978	Feature

Table 4.6: Example of how tasks are labeled

```
1 void findBug()
2   while inputFile != EOF //  EOF = End Of File
3     newLine = readLine()
4     if newLine.Type EQUALS 'Bug'
5       B = New Bug()
6       B.startDate = newLine.startDate
7       B.process = newLine.process
8       B.team = newLine.team
9       AddNewBug(B)
```

Listing 4.7: Pseudocode example of how bugs are found

4.3.1 Add Bug

When adding a new bug, each bug is gathered into a data structure based on which team the bug belongs to, as the tests in lines 2, 7, 12 and 17 of the listing 4.8 shows. After the right team is found the program tries to add the bug to the corresponding data structure as illustrated in lines 3, 8, 13 and 18. If the date of newly arrived bug already contains in the data structure, a counter representing the date is incremented and the new bug is discarded as shown the method dateExists in lines 26 to 32.

Since the program knows which team the new bug belongs to (after the checks on line 3, 8, 13 and 18) a counter can represent each bug. In the analyze the only important is to know how many bugs there are on a given date and which team it belongs to.

```
1  void addBug(Bug b)
2    if b.team EQUALS "TeamOne"
3      if dateExists(b.date, TeamOne) EQUALS false
4        // if date don't exists, then add the bug
5        TeamOne.add(b)
6
7    if b.team EQUALS "TeamTwo"
8      if dateExists(b.date, TeamTwo) EQUALS false
9        // if date don't exists, then add the bug
10       TeamTwo.add(b)
11
12   if b.team EQUALS "TeamThree"
13     if dateExists(b.date, TeamThree) EQUALS false
14       // if date don't exists, then add the bug
15       TeamThree.add(b)
16
17   if b.team EQUALS "TeamFour"
18     if dateExists(b.date, TeamFour) EQUALS false
19       // if date don't exists, then add the bug
20       TeamFour.add(b)
21
22 void dateExists(Date d, Arraylist list)
23   for Bug b in list
24     if b.date EQUALS d
25       b.counter++
26       return true
27
28 return false
29
```

Listing 4.8: Pseudocode example of how bugs are added

4.4 Throughput

Finding throughput per day is quite similar to how bugs are found (described in section 4.3). When reading in the data set a new throughput object is created for each line in the data set smilier to how the bugs was created. Then all throughput objects are sorted based on team association as shown in listing 4.8 . When all throughput objects are gathered, the program measure throughput. The throughput measurement is similar to the dateExists method (lines 26 to 32) in the listing 4.8 stated in section 4.3.1.

The dateExists method starts of with a test, the same test is done for bugs. If the date of the throughput object is in the data structure, the corresponding counter is incremented. If the date is not in the data structure, the new throughput object is added to the data structure. An excerpt of the code is listed in 4.9

```
1
2 dateExists(Throughput tp d, Arraylist list)
3   for Throughput t in list
4     if t.date EQUALS tp.date
5       t.counter++
6       return
7
8 structure.add(tp);
```

Listing 4.9: Pseudocode example of how throughput is measured

4.5 Churn

As stated in section 2.8 in order to take churn into account one need to know it's good surrogates. SI has gathered churn with help of Microsoft's Team Foundation Server (TFS). The TFS system automatically records data such as churn and lead time. Based on TFS one can know that churn for SI a is good surrogate.

To measure churn the data set from SI contains three columns ("Lines added, "Lines modified" and "Lines deleted") shown in table 4.7. These three columns are automatically measured by TFS. To complete the churn measure the three columns are multiplied. For task id one, the churn is 2028 ($352 + 307 + 1369 = 2028$). Some tasks has zero churn, for example task with id six, these tasks don't need code in order to be finished such tasks needs technical support to be finished.

Task id	Lines added	Lines modified	Lines deleted
1	352	307	1369
2	314	31	15
3	314	31	15
4	62	327	153
5	21	3	0
6	0	0	0

Table 4.7: How churn is presented in the excel document

4.6 Lead time

The program does not need to analyze the lead time for each task. The lead time for each task is stated in the data set as shown in table 4.8. The program will gather all the tasks that are started on the same day and belongs to the same team and add up their lead time together as shown in code listing 4.10.

ID	Type	Leadtime
84096	Feature	1
84118	Bug	25
84096	Feature	7
84118	Bug	13

Table 4.8: How lead time is recorded in the excel document

```
1
2 addLeadTime(lead_time t, Arraylist list)
3   for lead_time in list
4     if lead_time.date EQUALS t.date
5       lead_time.leadtime+= t.leadtime
6       return
7
8 structure.add(t)
```

Listing 4.10: Pseudocode example of lead time is measured

4.7 Lead time and churn

As in the article "Quantifying the Effect of Using Kanban versus Scrum" (Sjøberg, Johnsen and Solberg, 2012) to prevent outliers from having a large effect on the results,

the top and lowest ten percent of lead time and churn are removed from the data set.

Churn is removed because a module or a feature, which consists of hundred or thousand lines of code could be removed without much work and in this thesis churn is used as surrogate estimate the complexity of the task. Lead time is removed because some tasks could be given low priority due to lack of manpower in a given period or tasks could be labeled as not critical.

4.8 Outcome

This is how the result looks like after the program has measured WIP

4.9 SPSS

After the program has finished the measurements of the data, SPSS will be used to analyze the derived data. SPSS will help to answer the research question stated in chapter 1.2 with help of two statistics method: Correlation and case summaries.

"IBM® SPSS® Statistics is a comprehensive system for analyzing data. SPSS Statistics can take data from almost any type of file and use them to generate tabulated reports, charts and plots of distributions and trends, descriptive statistics, and complex statistical analyses." (IBM, 2014)

Chapter 5

Results

	T1	T2	T3	T4	T5	T6	T7
Throughput	0.43	0.43	0.43	0.43	0.43	0.43	0.43
Throughput Feature	0.46	0.46	0.46	0.46	0.46	0.46	0.46
Throughput bug	0.47	0.47	0.47	0.47	0.47	0.47	0.47
Bugs	0.27	0.27	0.27	0.27	0.27	0.27	0.27
Number of bugs finished in the same quarter	0.30	0.30	0.30	0.30	0.30	0.30	0.30
The average days for bugs in backlog	-0.18	-0.18	-0.18	-0.18	-0.18	-0.18	-0.18
Leadtime	0.10	0.10	0.10	0.10	0.10	0.10	0.10
Churn	-0.45	-0.45	-0.45	-0.45	-0.45	-0.45	-0.45
Churn for feature tasks	0.36	0.36	0.36	0.36	0.36	0.36	0.36
Churn for bug tasks	-0.45	-0.45	-0.45	-0.45	-0.45	-0.45	-0.45
The average churn	0.10	0.10	0.10	0.10	0.10	0.10	0.10

Table 5.1: Correlation - WIP
* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

5.1 Introduction

In this chapter

5.2 Team 1

5.2.1 WIP correlation

As shown in table 5.2 bugs, throughput, churn, churn bugs and throughput bugs have a significant correlation with WIP. In table 5.3 is descriptive statistic for WIP shown. As shown by the N column all quarters besides 2010-3 give reasonable data since in each quarter has 90 days give or take.

Correlation	Bugs	Throughput	Churn	Churn Bugs	Throughput bugs
WIP	.753*	.637**	.738*	.708*	.800**

Table 5.2: Team one - Correlation - WIP
* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
360	2010-3	39	5.82	5	2.55	15	1
	2010-4	92	1.7	2	0.70	4	1
	2011-1	90	4.37	2	6.87	31	1
	2011-2	91	14.25	5	14.497	52	3
	2011-3	92	2.4	3	0.97	4	1
	2011-4	92	14.26	4	22.718	97	1
	2012-1	91	13.35	5	16.62	67	4
	2012-2	91	15.35	3	27.145	94	2
	2012-3	92	16.97	13.5	8.719	52	5
	2012-4	90	9.07	2	17.073	74	1
	Total	860	9.99	4	16.04	97	1

Table 5.3: Descriptive statistics for WIP - Team one

5.2.1.1 WIP and throughput correlation

Both throughput and throughput bugs have a significant correlation relationship with WIP. This is inconsistent with Kanban philosophy and what the research says about WIP limit. All the research and the principles of Kanban suggest lowering WIP in order to get high throughput. In team one's case the higher the WIP the higher the throughput.

In table 5.3 the mean column shows indication that team one could have experimented with WIP limit the first five quarters. But in the last quarter, the WIP limit decreased. In the mean column in table 5.4 if one take a look at the quarter 2012-4, the throughput are second highest, the same quarter as WIP decreased. This link could have several reasons. One could be that they experimented with WIP and found out after the quarter 2012-3 that they needed to lower the WIP in order to increase the t hroughput, which it did. Or maybe team one didn't care about setting a WIP limit as stated in section 2.4.2 by Shinkle.

The throughput bugs variable and throughput variable both have a significant correlation with WIP, which make sense since throughput bugs are a subset of throughput. But throughput feature on the other hand has a -0.441 correlation with WIP. As shown in table 5.7. The correlation number -0.441 is not a significant correlation, but it's on the opposite site level of throughput and throughput feature although it's also a subset of throughput. This also goes in the literatures favor, because when WIP is low, more throughput feature is produced. Maybe this could have something to do with the relationship bugs and feature? Features are often more challenging and more fun to work with than bugs. One could also argue that maybe it's was significant fewer tasks labeled as bug than feature or vice versa, but as shown in table 5.7 that's not the case.

Team	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
360	2010-3	5	2.6	2	1.34	4	1
	2011-1	7	5.71	6	4.03	13	1
	2011-2	26	3.38	3	1.96	8	1
	2011-3	1	1	1	.	1	1
	2011-4	15	6.13	4	4.984	18	2
	2012-1	16	6.56	6.5	3.59	14	1
	2012-2	13	10.15	10	5.742	23	1
	2012-3	22	3.41	2	3.33	12	1
	2012-4	14	8.14	7	6.5	21	1
	Total	119	5.55	4	4.694	23	1

Table 5.4: Descriptive statistic for throughput - team one

Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
2010-3	14.00	3.07	2.00	3.47	14.00	1.00
2010-4	71.00	2.73	2.00	2.05	11.00	1.00
2011-1	67.00	2.55	2.00	1.96	13.00	1.00
2011-2	38.00	1.97	2.00	1.03	4.00	1.00
2011-3	43.00	1.65	1.00	0.92	4.00	1.00
2011-4	53.00	1.83	1.00	1.31	6.00	1.00
2012-1	40.00	1.90	1.00	1.46	7.00	1.00
2012-2	27.00	2.07	2.00	1.36	6.00	1.00
2012-3	40.00	2.70	2.00	2.43	13.00	1.00
2012-4	3.00	1.00	1.00	0	1.00	1.00
Total	396.00	2.26	2.00	1.82	14.00	1.00

Table 5.5: Descriptive statistic for throughput feature - team one

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
360	2010-3	9	1.11	1	0.33	2	1
	2010-4	3	1	1	0	1	1
	2011-2	26	5	3	4.45	17	1
	2011-3	1	1	1	.	1	1
	2011-4	24	6.79	4	6.35	20	1
	2012-1	18	4.83	4	3.20	11	1
	2012-2	19	7.84	9	5.843	17	1
	2012-3	6	2.67	1.5	2.25	6	1
	2012-4	11	3.18	3	1.77	7	1
	Total	117	5.08	3	4.88	20	1

Table 5.6: Descriptive statistic for throughput bugs - team one

Number of features	Number of bugs
660	594

Table 5.7: Number of throughput divided into bugs and feature - Team one

5.2.1.2 WIP and bugs correlation

Only looking at WIP and throughput correlation without taking other factores into account may give bias.

Bugs and churn have also have a significant correlation with WIP. A positive bug correlation could help favor what said about WIP and throughput in literature. Since both WIP and throughput increases and decreases at the same time, the same goes for bug. This could mean that when WIP is high, and throughput is high team one also has a lot of bugs in their code. This means that they don't do their job well, and this could be because they have a lot of task switching due to high WIP limit. As stated in table 5.8 one can see that almost each bug that was reported was finished in the same quarter. The outcast could be the bugs reported just before the quarter end. Worth noticing in quarter 2011-1 there was no bugs recorded. This conclude the statement that the WIP limit is set too high. 0.467

Quarter	Precent of bugs finished in the same quarter as reported
2010-3	100%
2010-4	100%
2011-1	0%
2011-2	97.74%
2011-3	20%
2011-4	97.5%
2012-1	96.51%
2012-2	98.63%
2012-3	82.35%
2012-4	100%

Table 5.8: The percent number of bugs finished in the same quarter as reported - Team one

5.2.1.3 WIP and churn correlation

Churn and WIP have a positive correlation. This means that when team one has more tasks in progress the tasks requires more coding. This is odd and could be a coincidence. Churn and churn bugs have also here as with throughput a close correlation with WIP. But churn feature has a correlation of 0.467

As shown in table 5.9 and the column 'maximum' the biggest feature task requires almost 60 times more code editing than the biggest bug task as showed in table 5.10. In the same table the column std. deviation shows way more proliferation for churn feature. This indicates that the numbers in churn feature are way more spread, which again indicates that it's random of how big the tasks are. In the churn bugs the std. deviation is much lower than in the feature table. This could explain why churn feature not has a positive correlation as churn and churn bugs. Both median and mean favors that churn feature requires more work than churn bugs

Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
2010-3	5	434	106	808.67	1879	30
2011-1	7	0	0	0	0	0
2011-2	26	1199.54	139	4695.424	24139	0
2011-3	1	0	0	.	0	0
2011-4	15	516.4	148	1063.92	3946	0
2012-1	16	5775.25	410	20692.09	83317	0
2012-2	14	7569.14	522.5	24655.70	93108	0
2012-3	23	6.74	0	25.44	118	0
2012-4	16	408	47	1056.23	4275	0
Total	123	2001.29	65	11380.06	93108	0

Table 5.9: Descriptive Statistic for churn feature - Team one

Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
2010-3	7	57.14	22	73.91	193	4
2010-4	2	30	30	41.012	59	1
2011-2	21	386.33	177	431.94	1604	2
2011-3	1	2	2	.	2	2
2011-4	19	306.89	148	437.13	1557	12
2012-1	16	312.81	160	307.73	1035	4
2012-2	13	296.92	169	292.29	1019	18
2012-3	7	37.71	9	56.494	126	0
2012-4	8	118.75	84	103.09	292	3
Total	94	260.48	135	346.25	1604	0

Table 5.10: Descriptive Statistic for churn bugs - Team one

5.2.2 Throughput

In table 5.11 is correlation for throughput for team one stated. Throughput has a positive correlation to bugs, WIP, churn, churn bugs, churn feature, throughput bugs and churn average. In table 5.12 is descriptive statistic for throughput shown. The range from 2011-2 to 2012-4 is the most valuable quarters based on the number of tasks finished if one excludes the quarter 2011-3.

Correlation	Bugs	WIP	Churn	Churn Bugs	Churn feature	Throughput bugs	Churn Average
Throughput	.945**	.816**	.962**	.776**	.736*	.940**	.638*

Table 5.11: Team one - Correlation - Throughput
* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
2010-3	5	2.6	2	1.34	4	1
2011-1	7	5.71	6	4.03	13	1
2011-2	26	3.38	3	1.96	8	1
2011-3	1	1	1	.	1	1
2011-4	15	6.13	4	4.984	18	2
2012-1	16	6.56	6.5	3.59	14	1
2012-2	13	10.15	10	5.742	23	1
2012-3	22	3.41	2	3.33	12	1
2012-4	14	8.14	7	6.5	21	1
Total	119	5.55	4	4.694	23	1

Table 5.12: Descriptive statistic for throughput - team one

5.2.2.1 Throughput and bugs correlation

The correlation between throughput and bugs is 0.945, which is significantly high. The quarters from 2011-2 to 2012-4 minus 2011-3 is also the most valid in bugs. The correlation between bugs and throughput means that when team one has high throughput of tasks they also produce more bugs. This correlation can be used to favor the literature of WIP as stated in section5.2.1. Since team one produces more bugs when they finished feature or bugs could be an indicator that they may have a too high WIP limit so they cannot focus enough on each task.

Throughput and bugs also have the same relationship as WIP and bugs, because throughput and throughput bugs has a significant correlation to both, but throughput

feature not. Throughput feature have correlation of -0.340 compared to -0.441 in WIP. What this means is that when team one has high throughput of tasks, they mostly produces tasks labeled as bug. This could be because the feature tasks are more comprehensive than the tasks labeled as bug so the lead time is longer for feature tasks than bug tasks. Or it could due the reason that when there is an upcoming release, all the bugs are pushed through the Kanban system.

In table 5.13 the hypotheses that feature tasks is more comprehensive is verified. Four out of the six valid quarters churn feature is significant higher. In table 5.14 one can see that throughput bug's lead time is higher than throughput feature in four out of six quarters. The lead time cannot be counted, since bugs usually have longer lead time than feature, because feature usually has higher priority than bugs.

Quarter	Churn feature	Churn bug
2010-3	0	29.3
2010-4	0	20
2011-1	0	0
2011-2	1199.54	59.14
2011-3	0	2
2011-4	516.4	76.66
2012-1	5775.25	1009.9
2012-2	7569.14	707.46
2012-3	6.74	15.68
2012-4	408	168.28

Table 5.13: team one - Average churn for feature and bugs - Bugs

Quarter	Throughput feature	Throughput bug
2010-3	23	51.33
2011-1	38.85	6
2011-2	19.73	25.46
2011-3	1	5
2011-4	30.6	36.38
2012-1	20.37	45.56
2012-2	35.30	55.89
2012-3	30.72	17.67
2012-4	30.57	8.82
Total	27.30	35.09

Table 5.14: Lead time for throughput feature and bugs per quarter

5.2.2.2 Throughput and churn correlation

Churn, churn bugs, churn feature and the average of churn all have a positive correlation with throughput which mean that when throughput is high, the tasks requires more coding. From table 5.15 in the number column one can see that in each quarter there is a lot of tasks. Although from the median column one can see that in 6 of the valid quarters, three of them have a median of 0, which gives and indicator that in these three quarter there is probably minority of tasks which has high churn. So the correlation between throughput and churn may not be valid.

Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
2010-3	11.00	17.00	11.00	18.94	60.00	0
2011-1	73.00	0	0	0	0	0
2011-2	136.00	14.40	3.00	21.51	83.00	0
2011-3	48.00	1.67	0	7.71	50.00	0
2011-4	156.00	14.81	6.50	19.91	86.00	0
2012-1	157.00	9.47	0	19.55	86.00	0
2012-2	238.00	10.11	0	18.50	83.00	0
2012-3	53.00	3.02	0	9.62	53.00	0
2012-4	26.00	21.42	14.50	20.55	67.00	1.00
Total	902.00	10.14	0	18.55	86.00	0

Table 5.15: Descriptive statistic for churn - team one

5.2.3 Bugs

Correlation	Throughput	WIP	Churn	Churn Bugs	Churn feature	TP bugs
Bugs	.945**	.753*	.992**	.813**	.730*	.984**

Table 5.16: Team two - Correlation - Throughput
* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

It's hard to find any correlation between lead time, WIP and throughput. In quarter three in 2011, lead time, WIP and throughput dropped to their lowest point. This was due to the summer vacation. In quarter two of 2011 and two of 2012 bugs dropped but both Throughput and WIP increased.

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
360	2010-3	7	1.29	1	0.48	2	1
	2010-4	4	1	1	0	1	1
	2011-2	32	4.16	3.5	3.63	14	1
	2011-3	5	1	1	0	1	1
	2011-4	26	6.15	4	5.80	22	1
	2012-1	21	4.09	4	2.7	11	1
	2012-2	17	8.59	9	5.59	19	1
	2012-3	7	2.43	2	1.512	5	1
	2012-4	11	3	3	2	6	1
	Total	131	4.53	3	4.44 t	22	1

Table 5.17: 360 - Descriptive statistic - Bugs

5.3 Team 2

5.3.1 WIP

For team two there is no significant correlation between WIP or any of the variables that have been measured. In table 5.19 one can see that every quarter beside 2010 is a valid quarter based on the number of dates in quarter. An excerpt of the correlation data is shown in table 5.18. As in section 5.2.1 for team one, throughput is fairly high, not significant high as for team one, but fairly high. For team two the correlation between bugs and WIP is quite low, so the conclusion stated in section 5.2.1 may not suit team two.

In table 5.20 the correlation for throughput for team two is shown. Here one can see that throughput and bugs have a significant correlation. That means when WIP increases, throughput increases and when throughput increases the number of bugs reported increases. Which could mean that the WIP limit is to high.

	Throughput	Bugs	Churn	Leadtime
WIP	0.341	0.278	-0.446	0.096

Table 5.18: Team two - Correlation - WIP
* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Frontend	2010-3	24	9.28	10	6.43	23	1
	2010-4	92	13.92	13	3.931	25	5
	2011-1	90	15.3	15.5	3.94	23	7
	2011-2	91	23.51	24	4.19	37	13
	2011-3	92	20.13	20	5.43	32	9
	2011-4	92	21.33	21	6.92	34	7
	2012-1	91	22.91	22	6.63	40	11
	2012-2	91	21.93	21	3.36	32	17
	2012-3	92	25.93	27	4.92	36	19
	2012-4	67	19.18	15	9.66	41	9
	Total	822	20.18	20	6.93	41	1

Table 5.19: Descriptive statistic - WIP - Team two

Correlation	Bugs	Throughput feature	Bugs finished in the same quarter
Throughput	.885**	.669*	.640*

Table 5.20: Team one - Correlation - Throughput
* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

5.3.2 Lead time

Correlation	Churn Bugs	Churn union	Precent bugs fininshed	Churn Average
Leadtime	-.771**	.704*	-.697*	.766**

Table 5.21: Team one - Correlation - Leadtime_union

Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
2010-3	14.00	3.07	2.00	3.47	14.00	1.00
2010-4	71.00	2.73	2.00	2.05	11.00	1.00
2011-1	67.00	2.55	2.00	1.96	13.00	1.00
2011-2	38.00	1.97	2.00	1.03	4.00	1.00
2011-3	43.00	1.65	1.00	0.92	4.00	1.00
2011-4	53.00	1.83	1.00	1.31	6.00	1.00
2012-1	40.00	1.90	1.00	1.46	7.00	1.00
2012-2	27.00	2.07	2.00	1.36	6.00	1.00
2012-3	40.00	2.70	2.00	2.43	13.00	1.00
2012-4	3.00	1.00	1.00	0	1.00	1.00
Total	396.00	2.26	2.00	1.82	14.00	1.00

Table 5.22: Descriptive statistic - Throughput - Team two

5.3.3 Analyzed data per quarter

In figure 5.1 the average WIP, Throughput, Lead time and Bugs per quarter for team 360 are shown. As stated in section 2.4.2, to get best possible throughput the WIP limit should be low. According to the figure 5.1, when WIP is high, throughput is also high and when WIP is low, throughput also low. This team data is contrary to what is suggested from Kanban and research.

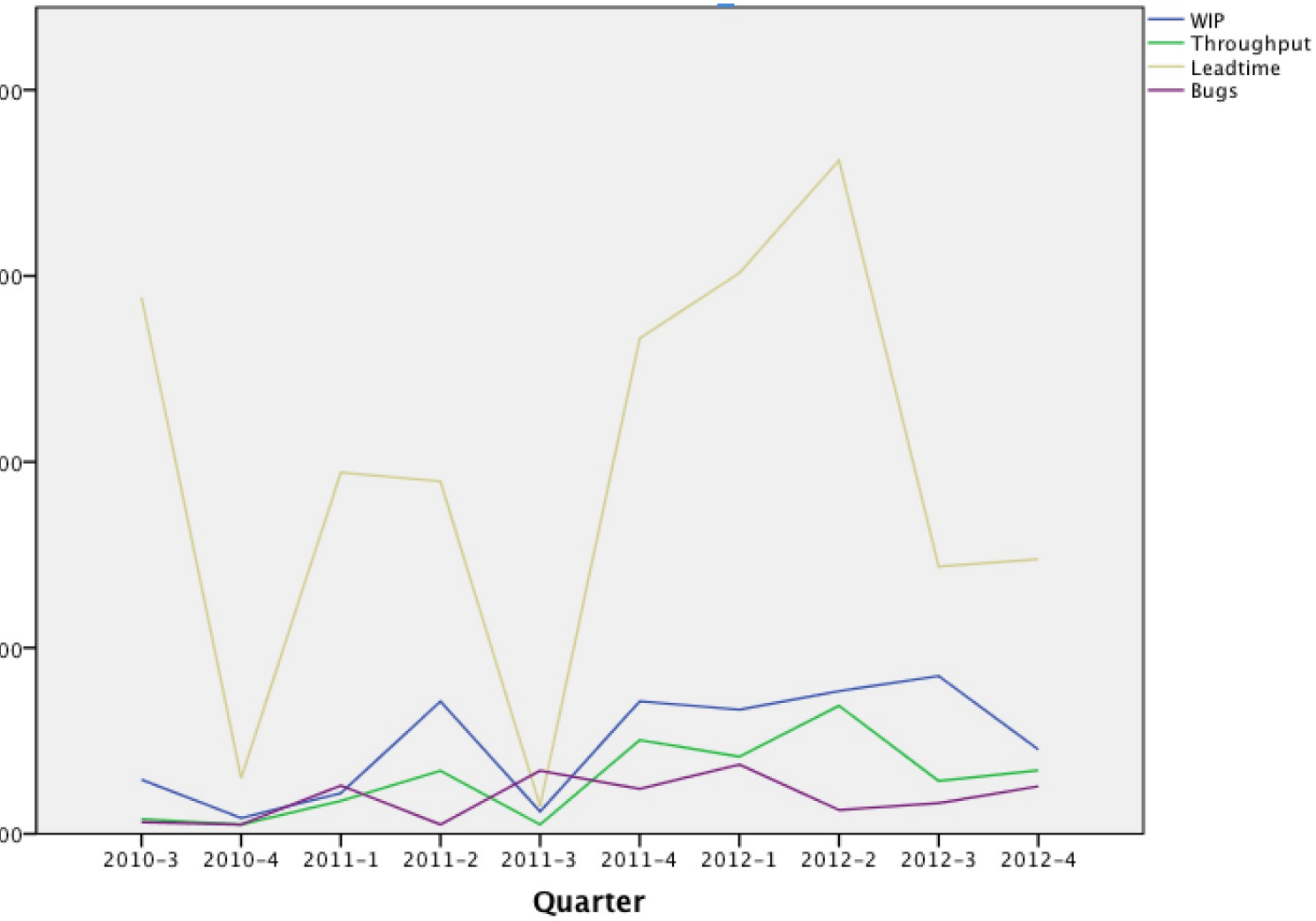


Figure 5.1: WIP, Throughput, Lead time and Bugs for 360 per quarter

In figure 5.2 the average WIP, Throughput, Lead time and Bugs per quarter for team Frontend are shown

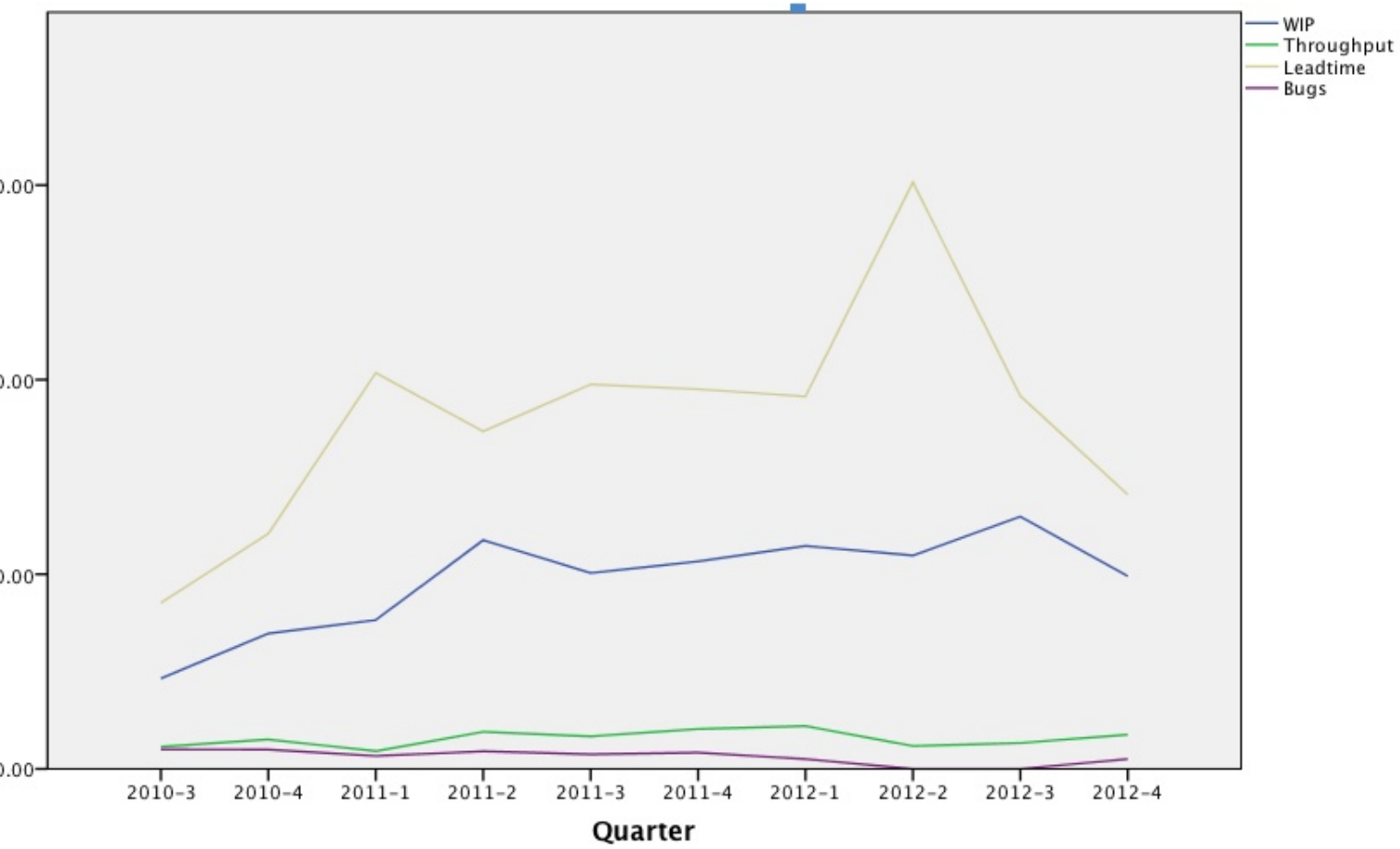


Figure 5.2: WIP, Throughput, Lead time and Bugs for Frontend per quarter

In figure 5.3 the average WIP, Throughput, Lead time and Bugs per quarter for team Krypton are shown

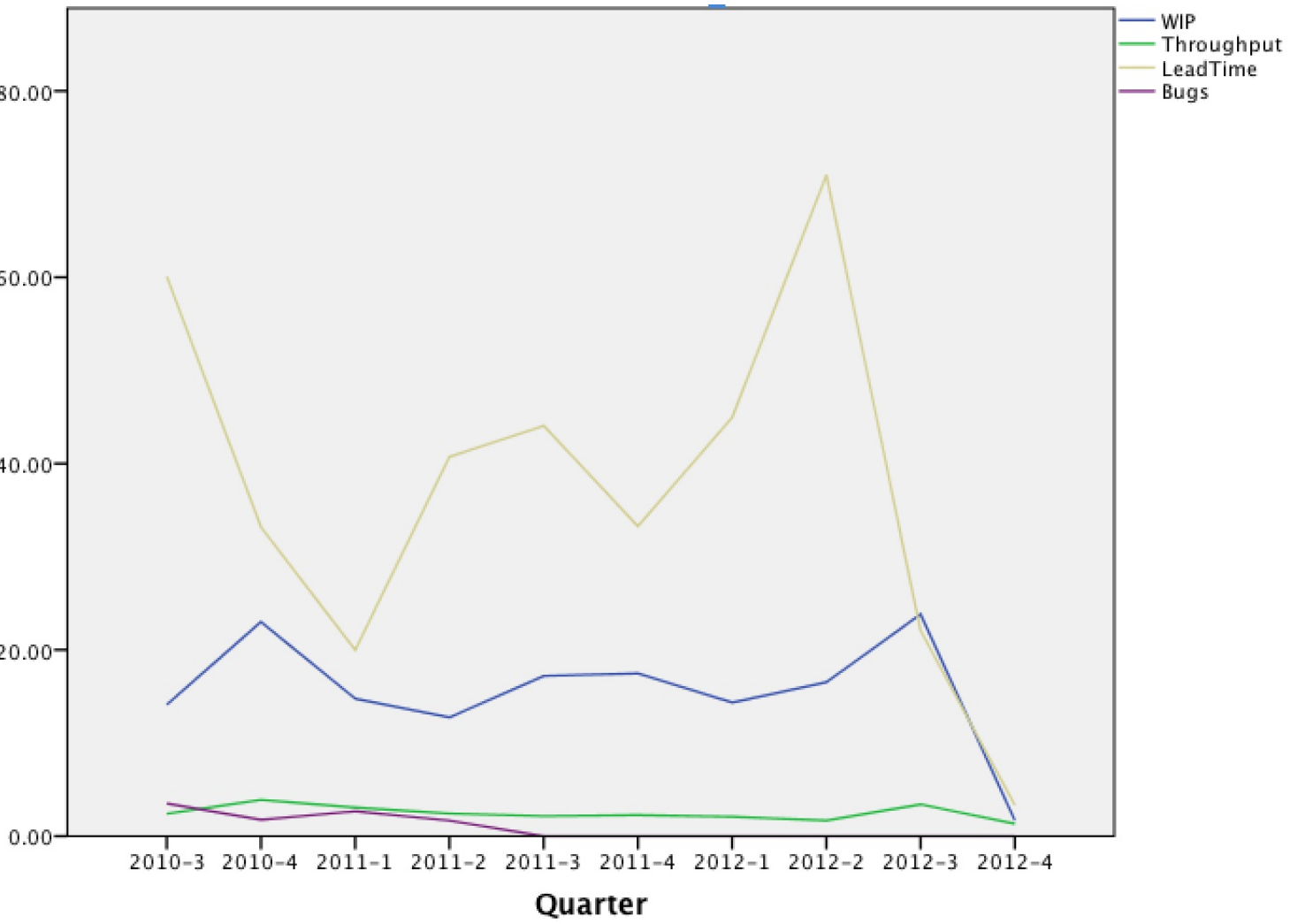
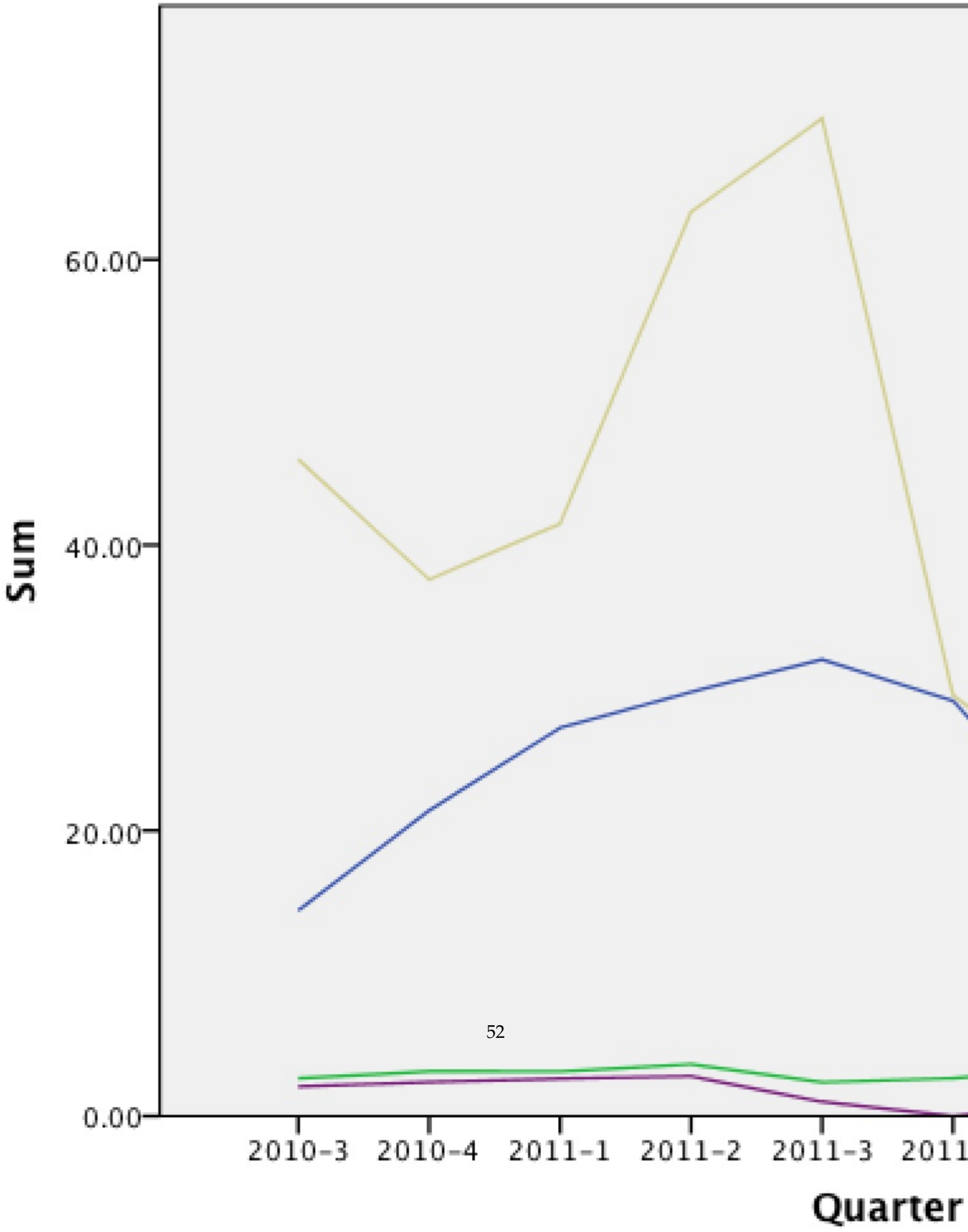


Figure 5.3: WIP, Throughput, Lead time and Bugs for Krypton per quarter

In figure 5.4 the average WIP, Throughput, Lead time and Bugs per quarter for team Neon are shown


60.00
40.00
20.00
0.00
Sum
2010-3
2010-4
2011-1
2011-2
2011-3
2011
Quarter

Til Dag

5.4 360

Correlations																	
		Bugs	Throughput	WIP	Churn	TP_Feature	Churn_Bugs	Leadtime_Union	Churn_Union	Churn_feature	TP_bugs	Average_Days_Backlog	Bugs_churn_average	Average_ft_churn	Precent_bugs_fininshed	Churn_Average
Bugs	Pearson Correlation	1	.945**	.753*	.992**	-0.328	.813**	-0.240	0.433	.730*	.984**	0.377	0.581	0.119	0.543	0.622
	Sig. (2-tailed)		0	0.012	0	0.354	0.004	0.505	0.212	0.026	0	0.282	0.078	0.744	0.104	0.055
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Throughput	Pearson Correlation	.945**	1	.816**	.962**	-0.340	.776**	-0.177	0.399	.736*	.940**	0.274	0.626	0.030	0.420	.638*
	Sig. (2-tailed)	0		0.004	0	0.336	0.008	0.624	0.253	0.024	0	0.443	0.053	0.934	0.227	0.047
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
WIP	Pearson Correlation	.753*	.816**	1	.738*	-0.441	.708*	-0.336	0.365	0.467	.800**	0.397	0.430	0.243	0.495	0.478
	Sig. (2-tailed)	0.012	0.004		0.015	0.202	0.022	0.342	0.299	0.205	0.005	0.255	0.215	0.498	0.146	0.163
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Churn	Pearson Correlation	.992**	.962**	.738*	1	-0.317	.804**	-0.203	0.372	.741*	.974**	0.349	0.581	0.087	0.475	0.617
	Sig. (2-tailed)	0	0	0.015		0.372	0.005	0.574	0.289	0.022	0	0.323	0.078	0.811	0.165	0.057
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
TP_Feature	Pearson Correlation	-0.328	-0.340	-0.441	-0.317	1	-0.419	0.376	-0.460	-0.208	-0.360	0.097	-0.171	-0.295	-.795**	-0.223
	Sig. (2-tailed)	0.354	0.336	0.202	0.372		0.228	0.284	0.181	0.591	0.307	0.789	0.637	0.408	0.006	0.536
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Churn_Bugs	Pearson Correlation	.813**	.776**	.708*	.804**	-0.419	1	-0.126	0.516	0.597	.884**	.700*	0.560	0.523	0.551	.665*
	Sig. (2-tailed)	0.004	0.008	0.022	0.005	0.228		0.729	0.127	0.090	0.001	0.024	0.092	0.121	0.099	0.036
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Leadtime_Union	Pearson Correlation	-0.240	-0.177	-0.336	-0.203	0.376	-0.126	1	-0.060	0.138	-0.280	-0.012	-0.003	0.158	-0.598	0.059
	Sig. (2-tailed)	0.505	0.624	0.342	0.574	0.284	0.729		0.868	0.724	0.433	0.974	0.994	0.663	0.068	0.871
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Churn_Union	Pearson Correlation	0.433	0.399	0.365	0.372	-0.460	0.516	-0.060	1	0.202	0.466	0.231	0.146	0.410	.634*	0.198
	Sig. (2-tailed)	0.212	0.253	0.299	0.289	0.181	0.127	0.868		0.603	0.175	0.522	0.687	0.239	0.049	0.584
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Churn_feature	Pearson Correlation	.730*	.736*	0.467	.741*	-0.208	0.597	0.138	0.202	1	.681*	0.073	.924**	0.008	0.362	.947**
	Sig. (2-tailed)	0.026	0.024	0.205	0.022	0.591	0.090	0.724	0.603		0.044	0.852	0	0.984	0.338	0
	N	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
TP_bugs	Pearson Correlation	.984**	.940**	.800**	.974**	-0.360	.884**	-0.280	0.466	.681*	1	0.494	0.574	0.192	0.568	0.620
	Sig. (2-tailed)	0	0	0.005	0	0.307	0.001	0.433	0.175	0.044		0.146	0.083	0.596	0.087	0.056
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Average_Days_Backlog	Pearson Correlation	0.377	0.274	0.397	0.349	0.097	.700*	-0.012	0.231	0.073	0.494	1	0.101	0.595	0.130	0.217
	Sig. (2-tailed)	0.282	0.443	0.255	0.323	0.789	0.024	0.974	0.522	0.852	0.146		0.782	0.070	0.721	0.548
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Bugs_churn_average	Pearson Correlation	0.581	0.626	0.430	0.581	-0.171	0.560	-0.003	0.146	.924**	0.574	0.101	1	-0.127	0.322	.973**
	Sig. (2-tailed)	0.078	0.053	0.215	0.078	0.637	0.092	0.994	0.687	0	0.083	0.782		0.726	0.365	0
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Average_ft_churn	Pearson Correlation	0.119	0.030	0.243	0.087	-0.295	0.523	0.158	0.410	0.008	0.192	0.595	-0.127	1	0.291	0.088
	Sig. (2-tailed)	0.744	0.934	0.498	0.811	0.408	0.121	0.663	0.239	0.984	0.596	0.070	0.726		0.415	0.808
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Precent_bugs_fininshed	Pearson Correlation	0.543	0.420	0.495	0.475	-.795**	0.551	-0.598	.634*	0.362	0.568	0.130	0.322	0.291	1	0.369
	Sig. (2-tailed)	0.104	0.227	0.146	0.165	0.006	0.099	0.068	0.049	0.338	0.087	0.721	0.365	0.415		0.294
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10
Churn_Average	Pearson Correlation	0.622	.638*	0.478	0.617	-0.223	.665*	0.059	0.198	.947**	0.620	0.217	.973**	0.088	0.369	1
	Sig. (2-tailed)	0.055	0.047	0.163	0.057	0.536	0.036	0.871	0.584	0	0.056	0.548	0	0.808	0.294	
	N	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10

** Correlation is significant at the 0.01 level (2-tailed).

* Correlation is significant at the 0.05 level (2-tailed).

Table 5.23: 360 - Correlation

Team	Quarter		Churn	Leadtime
360	2010-3	N	11	11
		Mean	17	3.55
		Median	11	3
		Std. Deviation	18.93	2.50
		Maximum	60	8
		Minimum	0	1
	2010-4	N	3	3
		Mean	0.67	1.67
		Median	1	2
		Std. Deviation	0.57	0.57
		Maximum	1	2
		Minimum	0	1
		N	136	136
	2011-2	Mean	14.4	2.7
		Median	3	2
		Std. Deviation	21.51	1.85
		Maximum	83	8
		Minimum	0	1
		N	48	48
	2011-3	Mean	1.67	3.06
		Median	0	2.5
		Std. Deviation	7.71	2.13
		Maximum	50	8
		Minimum	0	1
		N	156	156
	2011-4	Mean	14.81	2.25
		Median	6.5	2
		Std. Deviation	19.90	1.53
		Maximum	86	8
		Minimum	0	1
		N	157	157
	2012-1	Mean	9.47	2.68
		Median	0	2
		Std. Deviation	19.54	1.758
		Maximum	86	8
		Minimum	0	1
		N	238	238
	2012-2	Mean	10.11	2.78
		Median	0	2
		Std. Deviation	18.50	1.597
		Maximum	83	8
		Minimum	0	1
		N	53	53
	2012-3	Mean	3.02	1.68
		Median	0	1
		Std. Deviation	9.61	1.18
		Maximum	53	7
		Minimum	0	1
		N	26	26
	2012-4	Mean	21.42	2.08
		Median	14.5	2
		Std. Deviation	20.54	1.16
		Maximum	67	5
		Minimum	1	1
		N	902	902
	Total	Mean	10.14	2.7
		Median	0	2
		Std. Deviation	18.55	1.798
		Maximum	86	8
		Minimum	0	1

Table 5.24: Descriptive statistic - Lead-time and churn - 360

Frontend

Correlations																	
		WIP	Throughput	Bugs	Churn	TP_feature	Churn_Bugs	Churn_union	Leadtime_union	Churn_feature	Tp_bugs	Average_Days_Backlog_Bugs	Bugs_Churn_average	Churn_feature_Average	Precent_bugs_fininshed	Churn_Average
WIP	Pearson Correlation	1	0.491	0.278	-0.446	0.380	-0.446	0.316	0.096	0.362	0.508	-0.178	0.257	0.337	0.367	0.223
	Sig. (2-tailed)		0.150	0.436	0.196	0.279	0.196	0.373	0.791	0.304	0.134	0.622	0.473	0.341	0.297	0.537
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Throughput	Pearson Correlation	0.491	1	.773**	-0.036	.741*	0.217	-0.494	-0.539	-0.433	.758*	0.055	0.259	-0.609	.745*	-0.575
	Sig. (2-tailed)	0.150		0.009	0.921	0.014	0.548	0.147	0.108	0.212	0.011	0.879	0.470	0.062	0.013	0.082
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Bugs	Pearson Correlation	0.278	.773**	1	0.259	0.534	0.398	-0.555	-0.581	-0.478	0.271	0.005	0.296	-0.571	0.498	-0.624
	Sig. (2-tailed)	0.436	0.009		0.471	0.111	0.254	0.096	0.078	0.162	0.449	0.988	0.407	0.085	0.143	0.054
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn	Pearson Correlation	-0.446	-0.036	0.259	1	-0.219	.791**	-0.522	-0.449	-0.517	-0.110	-0.174	0.172	-0.485	-0.014	-0.471
	Sig. (2-tailed)	0.196	0.921	0.471		0.543	0.006	0.122	0.192	0.126	0.762	0.631	0.634	0.155	0.970	0.170
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
TP_feature	Pearson Correlation	0.380	.741*	0.534	-0.219	1	0.214	-0.244	-0.312	-0.026	0.427	-0.256	-0.007	-0.294	.640*	-0.319
	Sig. (2-tailed)	0.279	0.014	0.111	0.543		0.553	0.497	0.381	0.943	0.219	0.475	0.984	0.409	0.046	0.369
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_Bugs	Pearson Correlation	-0.446	0.217	0.398	.791**	0.214	1	-.707*	-.771**	-0.460	0.040	-0.378	-0.035	-0.590	0.388	-.689*
	Sig. (2-tailed)	0.196	0.548	0.254	0.006	0.553		0.022	0.009	0.181	0.912	0.282	0.924	0.073	0.267	0.027
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_union	Pearson Correlation	0.316	-0.494	-0.555	-0.522	-0.244	-.707*	1	.704*	.844**	-0.368	-0.139	0.152	.841**	-0.312	.981**
	Sig. (2-tailed)	0.373	0.147	0.096	0.122	0.497	0.022		0.023	0.002	0.296	0.702	0.674	0.002	0.379	0
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Leadtime_union	Pearson Correlation	0.096	-0.539	-0.581	-0.449	-0.312	-.771**	.704*	1	0.492	-0.301	0.327	-0.227	0.631	-.697*	.766**
	Sig. (2-tailed)	0.791	0.108	0.078	0.192	0.381	0.009	0.023		0.148	0.398	0.357	0.528	0.050	0.025	0.010
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_feature	Pearson Correlation	0.362	-0.433	-0.478	-0.517	-0.026	-0.460	.844**	0.492	1	-0.314	-0.358	-0.201	.921**	-0.172	.842**
	Sig. (2-tailed)	0.304	0.212	0.162	0.126	0.943	0.181	0.002	0.148		0.377	0.310	0.579	0	0.635	0.002
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Tp_bugs	Pearson Correlation	0.508	.758*	0.271	-0.110	0.427	0.040	-0.368	-0.301	-0.314	1	0.096	0.034	-0.393	0.622	-0.406
	Sig. (2-tailed)	0.134	0.011	0.449	0.762	0.219	0.912	0.296	0.398	0.377		0.792	0.926	0.261	0.055	0.245
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Average_Days_Backlog_Bugs	Pearson Correlation	-0.178	0.055	0.005	-0.174	-0.256	-0.378	-0.139	0.327	-0.358	0.096	1	-0.267	-0.272	-0.489	-0.035
	Sig. (2-tailed)	0.622	0.879	0.988	0.631	0.475	0.282	0.702	0.357	0.310	0.792		0.456	0.446	0.151	0.925
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Bugs_Churn_average	Pearson Correlation	0.257	0.259	0.296	0.172	-0.007	-0.035	0.152	-0.227	-0.201	0.034	-0.267	1	-0.197	0.371	0.016
	Sig. (2-tailed)	0.473	0.470	0.407	0.634	0.984	0.924	0.674	0.528	0.579	0.926	0.456		0.585	0.291	0.966
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_feature_Average	Pearson Correlation	0.337	-0.609	-0.571	-0.485	-0.294	-0.590	.841**	0.631	.921**	-0.393	-0.272	-0.197	1	-0.389	.846**
	Sig. (2-tailed)	0.341	0.062	0.085	0.155	0.409	0.073	0.002	0.050	0	0.261	0.446	0.585		0.267	0.002
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Precent_bugs_fininshed	Pearson Correlation	0.367	.745*	0.498	-0.014	.640*	0.388	-0.312	-.697*	-0.172	0.622	-0.489	0.371	-0.389	1	-0.445
	Sig. (2-tailed)	0.297	0.013	0.143	0.970	0.046	0.267	0.379	0.025	0.635	0.055	0.151	0.291	0.267		0.197
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_Average	Pearson Correlation	0.223	-0.575	-0.624	-0.471	-0.319	-.689*	.981**	.766**	.842**	-0.406	-0.035	0.016	.846**	-0.445	1
	Sig. (2-tailed)	0.537	0.082	0.054	0.170	0.369	0.027	0	0.010	0.002	0.245	0.925	0.966	0.002	0.197	
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

** Correlation is significant at the 0.01 level (2-tailed).

* Correlation is significant at the 0.05 level (2-tailed).

Table 5.25: Frontend Correlation

Team	Quarter		Leadtime	Churn
Frontend	2010-3	N	58	58
		Mean	7.43	127.93
		Median	6	53.5
		Std. Deviation	5.567	187.011
		Maximum	22	1072
		Minimum	1	0
	2010-4	N	130	130
		Mean	10.32	158.72
		Median	4	14
		Std. Deviation	15.30	444.805
		Maximum	78	4308
		Minimum	1	0
	2011-1	N	95	95
		Mean	28.01	566.73
		Median	11	47
		Std. Deviation	52.183	3696.125
		Maximum	434	35917
		Minimum	1	0
	2011-2	N	132	132
		Mean	12.17	496.8
		Median	5	19
		Std. Deviation	23.548	3240.172
		Maximum	184	35956
		Minimum	1	0
	2011-3	N	146	146
		Mean	10.66	174.07
		Median	6	14
		Std. Deviation	12.577	655.578
		Maximum	79	5650
		Minimum	1	0
	2011-4	N	167	167
		Mean	9.85	110.98
		Median	7	3
		Std. Deviation	10.9	480.32
		Maximum	62	5672
		Minimum	1	0
	2012-1	N	188	188
		Mean	13.9	118.64
		Median	8	5.5
		Std. Deviation	21.92	568.03
		Maximum	150	7565
		Minimum	1	0
	2012-2	N	77	77
		Mean	22.53	849.42
		Median	17	15
		Std. Deviation	25.52	3575.50
		Maximum	106	27170
		Minimum	1	0
	2012-3	N	93	93
		Mean	13.76	458.92
		Median	8	25
		Std. Deviation	18.023	1272.32
		Maximum	107	8600
		Minimum	1	0
	2012-4	N	82	82
		Mean	8.8007	428.67
		Median	4	54
		Std. Deviation	14.978	1178.471
		Maximum	83	9368
		Minimum	1	0
	Total	N	1168	1168
		Mean	13.35	305.62
		Median	7	15
		Std. Deviation	23.15	1883.16
		Maximum	434	35956
		Minimum	1	0

Table 5.26: Descriptive statistic - Lead-time and churn - Frontend

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Frontend	2010-3	14	2.79	2	2.91	12	1
	2010-4	39	2	1	1.53	8	1
	2011-1	20	1.85	1	1.226	5	1
	2011-2	29	2.86	2	3.21	16	1
	2011-3	37	2.41	2	1.93	10	1
	2011-4	35	2.89	1	5.02	30	1
	2012-1	39	3.56	2	3.90	23	1
	2012-2	31	1.81	1	1.4	7	1
	2012-3	24	2.54	2	1.79	7	1
	2012-4	14	2.86	2	1.79	7	1
	Total	282	2.56	2	2.895	30	1

Table 5.27: Frontend - Descriptive statistic - Bugs

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Frontend	2010-3	8	1.25	1	0.46	2	1
	2010-4	35	1.63	1	0.877	4	1
	2011-1	32	1.53	1	0.67	3	1
	2011-2	31	1.74	2	0.89	4	1
	2011-3	31	1.97	2	1.25	6	1
	2011-4	28	1.93	1	1.94	11	1
	2012-1	22	2.18	1	2.01	7	1
	2012-2	23	1.48	1	1.123	6	1
	2012-3	21	1.48	1	0.873	4	1
	2012-4	19	2.47	2	1.679	8	1
	Total	250	1.78	1	1.294	11	1

Table 5.28: Frontend - Descriptive statistic - TP feature

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Frontend	2010-3	15	2	2	1.363	6	1
	2010-4	41	1.98	2	1.387	8	1
	2011-1	18	1.33	1	0.48	2	1
	2011-2	35	2.50	2	2.161	10	1
	2011-3	38	2.18	2	1.60	8	1
	2011-4	40	2.6	2	2.52	14	1
	2012-1	41	3.44	3	2.61	10	1
	2012-2	29	1.9	1	1.472	7	1
	2012-3	30	2.13	1.5	1.69	8	1
	2012-4	25	2.12	1	2.10	10	1
	Total	312	2.31	2	1.982	14	1

Table 5.29: Frontend - Descriptive statistic - TP bugs

5.5 Krypton

Correlations																	
		WIP	Throughput	Bugs	Churn	TP_feature	Churn_Bugs	Churn_union	Leadtime_union	Churn_feature	Tp_bugs	Average_Days_Backlog_Bugs	Bugs_Churn_average	Churn_feature_Average	Precent_bugs_fininshed	Chur
WIP	Pearson Correlation	1	0.491	0.278	-0.446	0.380	-0.446	0.316	0.096	0.362	0.508	-0.178	0.257	0.337	0.367	0.223
	Sig. (2-tailed)		0.150	0.436	0.196	0.279	0.196	0.373	0.791	0.304	0.134	0.622	0.473	0.341	0.297	0.537
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Throughput	Pearson Correlation	0.491	1	.773**	-0.036	.741*	0.217	-0.494	-0.539	-0.433	.758*	0.055	0.259	-0.609	.745*	-0.57
	Sig. (2-tailed)	0.150		0.009	0.921	0.014	0.548	0.147	0.108	0.212	0.011	0.879	0.470	0.062	0.013	0.082
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Bugs	Pearson Correlation	0.278	.773**	1	0.259	0.534	0.398	-0.555	-0.581	-0.478	0.271	0.005	0.296	-0.571	0.498	-0.62
	Sig. (2-tailed)	0.436	0.009		0.471	0.111	0.254	0.096	0.078	0.162	0.449	0.988	0.407	0.085	0.143	0.054
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn	Pearson Correlation	-0.446	-0.036	0.259	1	-0.219	.791**	-0.522	-0.449	-0.517	-0.110	-0.174	0.172	-0.485	-0.014	-0.47
	Sig. (2-tailed)	0.196	0.921	0.471		0.543	0.006	0.122	0.192	0.126	0.762	0.631	0.634	0.155	0.970	0.170
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
TP_feature	Pearson Correlation	0.380	.741*	0.534	-0.219	1	0.214	-0.244	-0.312	-0.026	0.427	-0.256	-0.007	-0.294	.640*	-0.31
	Sig. (2-tailed)	0.279	0.014	0.111	0.543		0.553	0.497	0.381	0.943	0.219	0.475	0.984	0.409	0.046	0.369
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_Bugs	Pearson Correlation	-0.446	0.217	0.398	.791**	0.214	1	-.707*	-.771**	-0.460	0.040	-0.378	-0.035	-0.590	0.388	-.689
	Sig. (2-tailed)	0.196	0.548	0.254	0.006	0.553		0.022	0.009	0.181	0.912	0.282	0.924	0.073	0.267	0.027
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_union	Pearson Correlation	0.316	-0.494	-0.555	-0.522	-0.244	-.707*	1	.704*	.844**	-0.368	-0.139	0.152	.841**	-0.312	.981*
	Sig. (2-tailed)	0.373	0.147	0.096	0.122	0.497	0.022		0.023	0.002	0.296	0.702	0.674	0.002	0.379	0
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Leadtime_union	Pearson Correlation	0.096	-0.539	-0.581	-0.449	-0.312	-.771**	.704*	1	0.492	-0.301	0.327	-0.227	0.631	-.697*	.766*
	Sig. (2-tailed)	0.791	0.108	0.078	0.192	0.381	0.009	0.023		0.148	0.398	0.357	0.528	0.050	0.025	0.010
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_feature	Pearson Correlation	0.362	-0.433	-0.478	-0.517	-0.026	-0.460	.844**	0.492	1	-0.314	-0.358	-0.201	.921**	-0.172	.842*
	Sig. (2-tailed)	0.304	0.212	0.162	0.126	0.943	0.181	0.002	0.148		0.377	0.310	0.579	0	0.635	0.002
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Tp_bugs	Pearson Correlation	0.508	.758*	0.271	-0.110	0.427	0.040	-0.368	-0.301	-0.314	1	0.096	0.034	-0.393	0.622	-0.40
	Sig. (2-tailed)	0.134	0.011	0.449	0.762	0.219	0.912	0.296	0.398	0.377		0.792	0.926	0.261	0.055	0.245
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Average_Days_Backlog_Bugs	Pearson Correlation	-0.178	0.055	0.005	-0.174	-0.256	-0.378	-0.139	0.327	-0.358	0.096	1	-0.267	-0.272	-0.489	-0.03
	Sig. (2-tailed)	0.622	0.879	0.988	0.631	0.475	0.282	0.702	0.357	0.310	0.792		0.456	0.446	0.151	0.925
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Bugs_Churn_average	Pearson Correlation	0.257	0.259	0.296	0.172	-0.007	-0.035	0.152	-0.227	-0.201	0.034	-0.267	1	-0.197	0.371	0.016
	Sig. (2-tailed)	0.473	0.470	0.407	0.634	0.984	0.924	0.674	0.528	0.579	0.926	0.456		0.585	0.291	0.966
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_feature_Average	Pearson Correlation	0.337	-0.609	-0.571	-0.485	-0.294	-0.590	.841**	0.631	.921**	-0.393	-0.272	-0.197	1	-0.389	.846*
	Sig. (2-tailed)	0.341	0.062	0.085	0.155	0.409	0.073	0.002	0.050	0	0.261	0.446	0.585		0.267	0.002
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Precent_bugs_fininshed	Pearson Correlation	0.367	.745*	0.498	-0.014	.640*	0.388	-0.312	-.697*	-0.172	0.622	-0.489	0.371	-0.389	1	-0.44
	Sig. (2-tailed)	0.297	0.013	0.143	0.970	0.046	0.267	0.379	0.025	0.635	0.055	0.151	0.291	0.267		0.197
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_Average	Pearson Correlation	0.223	-0.575	-0.624	-0.471	-0.319	-.689*	.981**	.766**	.842**	-0.406	-0.035	0.016	.846**	-0.445	1
	Sig. (2-tailed)	0.537	0.082	0.054	0.170	0.369	0.027	0	0.010	0.002	0.245	0.925	0.966	0.002	0.197	
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

*Correlation is significant at the 0.05 level (2-tailed).

Table 5.30: Krypton - correlation

Team	Quarter		Leadtime	Churn
Krypton	2010-3	N	46	46
		Mean	7.89	159.38
		Median	3.5	26.5
		Std. Deviation	11.651	206.584
		Maximum	71	647
		Minimum	1	0
	2010-4	N	139	139
		Mean	7.38	97.7
		Median	3	28
		Std. Deviation	12.88	145.84
		Maximum	85	700
		Minimum	1	0
	2011-1	N	153	153
		Mean	5.83	71.38
		Median	2	17
		Std. Deviation	7.383	130.40
		Maximum	32	726
		Minimum	1	0
	2011-2	N	51	51
		Mean	14.37	110.45
		Median	9	31
		Std. Deviation	16.09	165.053
		Maximum	90	719
		Minimum	1	0
	2011-3	N	49	49
		Mean	13.71	119.43
		Median	8	27
		Std. Deviation	15.97	159.39
		Maximum	88	604
		Minimum	1	1
	2011-4	N	66	66
		Mean	10.5	112.36
		Median	6	36
		Std. Deviation	11.975	164.935
		Maximum	57	675
		Minimum	1	1
	2012-1	N	45	45
		Mean	23.73	99.09
		Median	8	23
		Std. Deviation	40.47	149.02
		Maximum	180	636
		Minimum	1	1
	2012-2	N	45	45
		Mean	23	109.07
		Median	7	47
		Std. Deviation	32.08	131.58
		Maximum	135	536
		Minimum	1	1
	2012-3	N	74	74
		Mean	6.92	82.39
		Median	3	37
		Std. Deviation	9.26	122.637
		Maximum	35	713
		Minimum	1	1
	2012-4	N	2	2
		Mean	1.5	44
		Median	1.5	44
		Std. Deviation	0.70	1.41
		Maximum	2	45
		Minimum	1	43
	Total	N	683	683
		Mean	10.55	98.6
		Median	4	29
		Std. Deviation	18.1	149.01
		Maximum	180	726
		Minimum	1	0

Table 5.31: Descriptive statistic - Lead-time and churn - Krypton

Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
2010-3	14.00	3.07	2.00	3.47	14.00	1.00
2010-4	71.00	2.73	2.00	2.05	11.00	1.00
2011-1	67.00	2.55	2.00	1.96	13.00	1.00
2011-2	38.00	1.97	2.00	1.03	4.00	1.00
2011-3	43.00	1.65	1.00	0.92	4.00	1.00
2011-4	53.00	1.83	1.00	1.31	6.00	1.00
2012-1	40.00	1.90	1.00	1.46	7.00	1.00
2012-2	27.00	2.07	2.00	1.36	6.00	1.00
2012-3	40.00	2.70	2.00	2.43	13.00	1.00
2012-4	3.00	1.00	1.00	0	1.00	1.00
Total	396.00	2.26	2.00	1.82	14.00	1.00

Table 5.32: Frontend - Descriptive statistic - Throughput

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Krypton	2010-3	25	14.12	14	5.093	27	9
	2010-4	92	23	23.5	7.819	48	12
	2011-1	90	14.76	13.5	5	26	6
	2011-2	91	12.75	12	4.64	22	3
	2011-3	92	17.2	16	6.476	32	8
	2011-4	92	17.47	17	6.32	30	4
	2012-1	91	14.35	15	3.911	24	7
	2012-2	91	16.53	14	6.03	34	10
	2012-3	92	23.83	20	10.387	43	7
	2012-4	67	1.69	0	3.016	11	0
	Total	823	16.11	15	8.43	48	0

Table 5.33: Krypton - Descriptive statistic WIP

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Krypton	2010-3	17	1.76	2	0.83	3	1
	2010-4	28	1.71	1.5	0.85	4	1
	2011-1	39	3.1	2	2.51	12	1
	2011-2	26	2.04	2	1.28	5	1
	2011-3	25	1.64	1	1.14	5	1
	2011-4	26	2	2	1.35	7	1
	2012-1	26	1.85	1	1.617	9	1
	2012-2	19	2.52	2	2.48	11	1
	2012-3	31	2.16	2	1.44	6	1
	Total	241	2.12	2	1.69	12	1
	2010-3	20	2.65	2	3.54	17	1

Table 5.34: Krypton - Descriptive statistic - Bugs

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Krypton	2010-3	10	2.20	1.5	1.619	5	1
	2010-4	48	2.79	2	2.278	11	1
	2011-1	30	1.7	1	0.95	4	1
	2011-2	17	1.65	1	1.115	4	1
	2011-3	20	1.45	1	0.82	4	1
	2011-4	31	1.48	1	0.89	4	1
	2012-1	14	1.71	1	1.139	5	1
	2012-2	11	1.73	1	0.90	3	1
	2012-3	17	1.53	1	0.8	4	1
	2012-4	3	1	1	0	1	1
	Total	201	1.9	1	1.48	11	1

Table 5.35: Krypton - Descriptive statistic - TP feature

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Krypton	2010-3	6	3.5	2.5	3.2090	9	1
	2010-4	34	1.76	2	0.89	4	1
	2011-1	45	2.67	2	2.20	13	1
	2011-2	28	1.68	2	0.77	4	1
	2011-3	26	1.62	1	0.752	3	1
	2011-4	27	1.89	1	1.45	6	1
	2012-1	28	1.86	1	1.407	7	1
	2012-2	18	2.06	2	1.552	6	1
	2012-3	28	2.93	3	2.19	9	1
	Total	240	2.13	2	1.66	13	1

Table 5.36: Krypton - Descriptive statistic - TP bugs

5.6 Neon

Correlations																	
		WIP	Throughput	Bugs	Churn	TP_feature	Churn_Bugs	Churn_union	Leadtime_union	Churn_feature	Tp_bugs	Average_Days_Backlog_Bugs	Bugs_Churn_average	Churn_feature_Average	Precent_bugs_fininshed	Churn
WIP	Pearson Correlation	1	0.491	0.278	-0.446	0.380	-0.446	0.316	0.096	0.362	0.508	-0.178	0.257	0.337	0.367	0.223
	Sig. (2-tailed)		0.150	0.436	0.196	0.279	0.196	0.373	0.791	0.304	0.134	0.622	0.473	0.341	0.297	0.537
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Throughput	Pearson Correlation	0.491	1	.773**	-0.036	.741*	0.217	-0.494	-0.539	-0.433	.758*	0.055	0.259	-0.609	.745*	-0.575
	Sig. (2-tailed)	0.150		0.009	0.921	0.014	0.548	0.147	0.108	0.212	0.011	0.879	0.470	0.062	0.013	0.082
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Bugs	Pearson Correlation	0.278	.773**	1	0.259	0.534	0.398	-0.555	-0.581	-0.478	0.271	0.005	0.296	-0.571	0.498	-0.624
	Sig. (2-tailed)	0.436	0.009		0.471	0.111	0.254	0.096	0.078	0.162	0.449	0.988	0.407	0.085	0.143	0.054
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn	Pearson Correlation	-0.446	-0.036	0.259	1	-0.219	.791**	-0.522	-0.449	-0.517	-0.110	-0.174	0.172	-0.485	-0.014	-0.471
	Sig. (2-tailed)	0.196	0.921	0.471		0.543	0.006	0.122	0.192	0.126	0.762	0.631	0.634	0.155	0.970	0.170
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
TP_feature	Pearson Correlation	0.380	.741*	0.534	-0.219	1	0.214	-0.244	-0.312	-0.026	0.427	-0.256	-0.007	-0.294	.640*	-0.319
	Sig. (2-tailed)	0.279	0.014	0.111	0.543		0.553	0.497	0.381	0.943	0.219	0.475	0.984	0.409	0.046	0.369
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_Bugs	Pearson Correlation	-0.446	0.217	0.398	.791**	0.214	1	-.707*	-.771**	-0.460	0.040	-0.378	-0.035	-0.590	0.388	-.689*
	Sig. (2-tailed)	0.196	0.548	0.254	0.006	0.553		0.022	0.009	0.181	0.912	0.282	0.924	0.073	0.267	0.027
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_union	Pearson Correlation	0.316	-0.494	-0.555	-0.522	-0.244	-.707*	1	.704*	.844**	-0.368	-0.139	0.152	.841**	-0.312	.981**
	Sig. (2-tailed)	0.373	0.147	0.096	0.122	0.497	0.022		0.023	0.002	0.296	0.702	0.674	0.002	0.379	0
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Leadtime_union	Pearson Correlation	0.096	-0.539	-0.581	-0.449	-0.312	-.771**	.704*	1	0.492	-0.301	0.327	-0.227	0.631	-.697*	.766**
	Sig. (2-tailed)	0.791	0.108	0.078	0.192	0.381	0.009	0.023		0.148	0.398	0.357	0.528	0.050	0.025	0.010
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_feature	Pearson Correlation	0.362	-0.433	-0.478	-0.517	-0.026	-0.460	.844**	0.492	1	-0.314	-0.358	-0.201	.921**	-0.172	.842**
	Sig. (2-tailed)	0.304	0.212	0.162	0.126	0.943	0.181	0.002	0.148		0.377	0.310	0.579	0	0.635	0.002
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Tp_bugs	Pearson Correlation	0.508	.758*	0.271	-0.110	0.427	0.040	-0.368	-0.301	-0.314	1	0.096	0.034	-0.393	0.622	-0.406
	Sig. (2-tailed)	0.134	0.011	0.449	0.762	0.219	0.912	0.296	0.398	0.377		0.792	0.926	0.261	0.055	0.245
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Average_Days_Backlog_Bugs	Pearson Correlation	-0.178	0.055	0.005	-0.174	-0.256	-0.378	-0.139	0.327	-0.358	0.096	1	-0.267	-0.272	-0.489	-0.035
	Sig. (2-tailed)	0.622	0.879	0.988	0.631	0.475	0.282	0.702	0.357	0.310	0.792		0.456	0.446	0.151	0.925
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Bugs_Churn_average	Pearson Correlation	0.257	0.259	0.296	0.172	-0.007	-0.035	0.152	-0.227	-0.201	0.034	-0.267	1	-0.197	0.371	0.016
	Sig. (2-tailed)	0.473	0.470	0.407	0.634	0.984	0.924	0.674	0.528	0.579	0.926	0.456		0.585	0.291	0.966
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_feature_Average	Pearson Correlation	0.337	-0.609	-0.571	-0.485	-0.294	-0.590	.841**	0.631	.921**	-0.393	-0.272	-0.197	1	-0.389	.846**
	Sig. (2-tailed)	0.341	0.062	0.085	0.155	0.409	0.073	0.002	0.050	0	0.261	0.446	0.585		0.267	0.002
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Precent_bugs_fininshed	Pearson Correlation	0.367	.745*	0.498	-0.014	.640*	0.388	-0.312	-.697*	-0.172	0.622	-0.489	0.371	-0.389	1	-0.445
	Sig. (2-tailed)	0.297	0.013	0.143	0.970	0.046	0.267	0.379	0.025	0.635	0.055	0.151	0.291	0.267		0.197
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Churn_Average	Pearson Correlation	0.223	-0.575	-0.624	-0.471	-0.319	-.689*	.981**	.766**	.842**	-0.406	-0.035	0.016	.846**	-0.445	1
	Sig. (2-tailed)	0.537	0.082	0.054	0.170	0.369	0.027	0	0.010	0.002	0.245	0.925	0.966	0.002	0.197	
	N	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

* Correlation is significant at the 0.05 level (2-tailed).

** Correlation is significant at the 0.01 level (2-tailed).

Table 5.37: Neon - Correlation

Team	Quarter		Churn	Leadtime
Neon	2010-3	N	62	62
		Mean	25.27	7.69
		Median	9	6
		Std. Deviation	39.448	6.88
		Maximum	193	24
		Minimum	0	1
	2010-4	N	125	125
		Mean	33.13	6.66
		Median	15	5
		Std. Deviation	47.46	6.01
		Maximum	214	27
		Minimum	0	1
	2011-1	N	132	132
		Mean	23.81	6.53
		Median	9	5.5
		Std. Deviation	41.22	5.43
		Maximum	301	27
		Minimum	0	1
	2011-2	N	134	134
		Mean	36.1	10.71
		Median	7	6
		Std. Deviation	59.067	35.19
		Maximum	271	408
		Minimum	0	1
	2011-3	N	99	99
		Mean	40.86	8.34
		Median	17	6
		Std. Deviation	50.60	6.61
		Maximum	239	27
		Minimum	1	1
	2011-4	N	98	98
		Mean	63.31	8.5
		Median	38	7
		Std. Deviation	74.35	6.84
		Maximum	294	27
		Minimum	1	1
	2012-1	N	110	110
		Mean	56.75	5.16
		Median	21	4
		Std. Deviation	72.52	4.46
		Maximum	296	22
		Minimum	1	1
	2012-2	N	81	81
		Mean	44.95	5.05
		Median	22	4
		Std. Deviation	54.23	4.43
		Maximum	300	23
		Minimum	1	1
	2012-3	N	174	174
		Mean	66.12	5.78
		Median	33.5	4
		Std. Deviation	81.02	4.35
		Maximum	299	21
		Minimum	1	1
	2012-4	N	120	120
		Mean	75.930000000000007	5.3
		Median	32	4.5
		Std. Deviation	86.49	4.
		Maximum	303	17
		Minimum	1	1
	Total	N	1155	1155
		Mean	47.76	7
		Median	18	5
		Std. Deviation	66.37	13.112
		Maximum	303	408
		Minimum	0	1

Table 5.38: Descriptive statistic - Lead-time and churn - Neon

Team	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Neon	2010-3	10	2	1.5	1.333	5	1
	2010-4	34	1.68	1	1.173	6	1
	2011-1	31	1.71	1	1.03	5	1
	2011-2	6	1.17	1	0.4	2	1
	2011-3	9	1	1	0	1	1
	2011-4	1	1	1	.	1	1
	Total	91	1.62	1	1.06	6	1

Table 5.39: Neon - Descriptive statistic - Throughput

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Neon	2010-3	25	14.4	15	6.19	23	6
	2010-4	92	21.41	20	7.17	41	9
	2011-1	90	27.2	27.5	4.90	38	17
	2011-2	91	29.71	27	14.42	62	12
	2011-3	92	31.98	30	8.90	55	18
	2011-4	92	29.09	29	10.09	45	12
	2012-1	91	19.03	18	4.63	30	7
	2012-2	91	24.34	25	10.282	50	5
	2012-3	92	23.24	21.5	7.89	44	10
	2012-4	92	19.48	22	10.99	45	1
	2013-1	11	2.18	2	0.60	3	1
	Total	859	24.45	24	10.54	62	1

Table 5.40: Neon - Descriptive statistic - WIP

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Neon	2010-4	40	2.45	2	1.694	9	1
	2011-1	47	2.43	2	1.80	8	1
	2011-2	42	3.57	3	2.52	13	1
	2011-3	45	2.47	2	2.33	13	1
	2011-4	48	2.48	2	1.59	7	1
	2012-1	36	3.25	3	3.03	16	1
	2012-2	36	2.19	2	1.48	8	1
	2012-3	44	3.43	2	2.55	10	1
	2012-4	33	3.12	2	2.63	10	1
	2013-1	1	1	1	.	1	1
	Total	404	2.75	2	2.306	17	1

Table 5.41: Neon - Descriptive statistic - Bugs

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Neon	2010-3	10	2	1.5	1.333	5	1
	2010-4	34	1.68	1	1.173	6	1
	2011-1	31	1.71	1	1.038	5	1
	2011-2	11	1.27	1	0.64	3	1
	2011-3	34	1.53	1	0.99	5	1
	2011-4	16	1.5	1	0.73	3	1
	2012-1	23	1.3	1	0.70	4	1
	2012-2	32	1.63	1	0.83	4	1
	2012-3	32	2.02	2	0.93	4	1
	2012-4	40	1.93	2	1.11	6	1
	Total	263	1.69	1	0.997	6	1

Table 5.42: Neon - Descriptive statistic - TP feature

TeamName	Quarter	N	Mean	Median	Std. Deviation	Maximum	Minimum
Neon	2010-3	10	2	1.5	1.333	5	1
	2010-4	34	1.68	1	1.173	6	1
	2011-1	31	1.71	1	1.038	5	1
	2011-2	11	1.27	1	0.64	3	1
	2011-3	34	1.53	1	0.99	5	1
	2011-4	16	1.5	1	0.73	3	1
	2012-1	23	1.3	1	0.70	4	1
	2012-2	32	1.63	1	0.83	4	1
	2012-3	32	2.02	2	0.93	4	1
	2012-4	40	1.93	2	1.11	6	1
	Total	263	1.69	1	0.997	6	1

Table 5.43: Neon - Descriptive statistic - TP bugs

Chapter 6

Discussion

Chapter 7

Conclusion

Bibliography

Adams, M. and B. Smoak (1990). 'Managing manufacturing improvement using computer integrated manufacturing methods'. In: *Semiconductor Manufacturing Science Symposium, 1990. ISMSS 1990., IEEE/SEMI International*, pp. 9–13. DOI: 10.1109/ISMSS.1990.66111.

Alliance, Scrum (2012). 'Scrum, A description'. In:

Anderson, David J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press. ISBN: 0984521402.

Anderson, David et al. (2011). 'Studying Lean-Kanban Approach Using Software Process Simulation'. In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by Alberto Sillitti et al. Vol. 77. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 12–26. ISBN: 978-3-642-20676-4. DOI: 10.1007/978-3-642-20677-1_2.

Beedle, Mike et al. (1999). 'SCRUM: An extension pattern language for hyperproductive software development'. In: *Pattern Languages of Program Design* 4, pp. 637–651.

Brekkan, Elin and Eystein Mathisen (2010). 'Introducing Scrum in Companies in Norway: A Case Study'. In:

Conboy, Kieran (2009). 'Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development'. In: *Information Systems Research* 20.3, pp. 329–354. DOI: 10.1287/isre.1090.0236.

Concas, Giulio et al. (2013). 'Simulation of software maintenance process, with and without a work-in-process limit'. In: *Journal of Software: Evolution and Process* 25.12, pp. 1225–1248. ISSN: 2047-7481. DOI: 10.1002/smr.1599.

El-Emam, Khaled (2000). 'A methodology for validating software product metrics'. In:

Gandomani, TAGHI JAVDANI et al. (2013). 'Important considerations for agile software development methods governance.' In: *Journal of Theoretical & Applied Information Technology* 55.3.

Gerring, John (2006). *Case Study Research: Principles and Practices*. Cambridge University Press. ISBN: 0521676568.

Gupta, Vikram (May 2013). *InfoQ Interviews David J. Anderson at Lean Kanban 2013 Conference*. URL: http://www.infoq.com/articles/David_Anderson_Lean_Kanban_2013_Conference_Intervew (visited on 01/10/2013).

IBM (Jan. 2014). *IBM SPSS Statistics 21 Core System User's Guide*. URL: http://www.sussex.ac.uk/its/pdfs/SPSS_Core_System_Users_Guide_21.pdf (visited on 30/01/2014).

Ikonen, Marko et al. (2010). 'Exploring the sources of waste in Kanban software development projects'. In: *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*. IEEE, pp. 376–381.

Ikonen, M. et al. (2011). 'On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation'. In: *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on*, pp. 305–314. DOI: 10.1109/ICECCS.2011.37.

Investopedia (30th Nov. 2013). *Quarter - Q1, Q2, Q3, Q4*. URL: http://www.investopedia.com/terms/q/quarter.asp (visited on 30/11/2013).

JD (18th Dec. 2013). *Pseudocode Standard*. URL: http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html (visited on 18/12/2013).

Kniberg, Henrik (2010). *Kanban and Scrum - making the most of both*. lulu.com. ISBN: 0557138329.

Lai, C.L., W.B. Lee and W.H. Ip (2003). 'A study of system dynamics in just-in-time logistics'. In: 138, pp. 265–269.

Leonardo Campos Rafael Buzon, Eric Fer (Mar. 2013). *Kanban Pioneer: Interview with David J. Anderson*. URL: http://www.infoq.com/articles/David-Anderson-Kanban/ (visited on 30/09/2013).

Mark L. Spearman, David L. Woodruff and Wallace J.Hopp (1990). 'CONWIP: a pull alternative to kanban'. In: 28.5, pp. 879–894.

Middleton, P. and D. Joyce (2012). 'Lean Software Management: BBC Worldwide Case Study'. In: *Engineering Management, IEEE Transactions on* 59.1, pp. 20–32. ISSN: 0018-9391. DOI: 10.1109/TEM.2010.2081675.

Munassar, Nabil Mohammed Ali and A Govardhan (2010). 'A Comparison Between Five Models Of Software Engineering.' In: *International Journal of Computer Science Issues (IJCSI)* 7.5.

Ohno, Taiichi (2001). *Toyota Production System on Compact Disc: Beyond Large-Scale Production*. Productivity Press. ISBN: 1563272679.

Oracle (2013). 'Arraylist'. In:

Poppendieck, Mary (2003). 'Lean Development and the Predictability Paradox'. In:

Poppendieck, Mary and Tom Poppendieck (2003). *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional. ISBN: 0321150783.

– (2006). *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Professional. ISBN: 0321437381.

– (2009). *Leading Lean Software Development: Results Are not the Point*. Addison-Wesley Professional. ISBN: 0321620704.

Raman, S. (Oct. 1998). 'Lean software development: is it feasible?' In: *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE*. Vol. 1, C13/1–C13/8 vol.1. DOI: 10.1109/DASC.1998.741480.

Rouse, Margaret (2005). *Throughput*. URL: http://searchnetworking.techtarget.com/definition/throughput (visited on 04/03/2014).

Seikola, M., H. Loisa and A. Jagos (Aug. 2011). 'Kanban Implementation in a Telecom Product Maintenance'. In: *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pp. 321–329. DOI: 10.1109/SEAA.2011.56.

Shepard, Jon M. and Robert W. Greene (2002). *Sociology and You*. Glencoe/McGraw-Hill. ISBN: 0078285763.

Shinkle, C.M. (2009). 'Applying the Dreyfus Model of Skill Acquisition to the Adoption of Kanban Systems at Software Engineering Professionals (SEP)'. In: *Agile Conference, 2009. AGILE '09.* Pp. 186–191. DOI: 10.1109/AGILE.2009.25.

Sienkiewicz, Lukasz (2012). 'Scrumban - the Kanban as an addition to Scrum software development method in a Network Organization'. In:

Sjøberg, D.I.K., A. Johnsen and J. Solberg (2012). 'Quantifying the Effect of Using Kanban versus Scrum: A Case Study'. In: *Software, IEEE* 29.5, pp. 47–53. ISSN: 0740-7459. DOI: 10.1109/MS.2012.110.

Software Innovation (Dec. 2013). URL: http://www.software-innovation.com/EN/COMPANY/pages/default_.aspx (visited on 12/12/2013).

Srinivasan, Mandyam M., Steven J. Ebbing and Alan T. Swearingen (2003). 'Woodward Aircraft Engine Systems Sets Work-in-Process Levels for High-Variety, Low-Volume Products'. In: *Interfaces* 33.4, pp. 61–69. DOI: 10.1287/inte.33.4.61.16377.

Yamashita, A, B Anda, A Mockus et al. (2012). 'Quantifying the effect of code smells on maintenance effort'. In:

Yin, Robert K. (2008). *Case Study Research: Design and Methods (Applied Social Research Methods)*. SAGE Publications, Inc. ISBN: 1412960991.